,

VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Analýza Temporálních Sítí

Temporal Network Analysis

2017

David Podermański

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Zadání diplomové práce

Student: **Bc. David Podermański**

Studijní program: N2647 Informační a komunikační technologie

Studijní obor: 2612T025 Informatika a výpočetní technika

Téma: Analýza temporálních sítí
Temporal Network Analysis

Jazyk vypracování: čeština

Zásady pro vypracování:

Cílem diplomové práce je analýza topologických vlastností temporálních sítí. Mnoho reálných systémů, od technologických přes společenské až po přírodní, lze modelovat sítěmi (grafy), tedy vrcholy, které jsou spojeny hranami. Řadu jejich vlastností, globálních či lokálních, můžeme určit jen na základě topologie sítě. Zohledněním netopologických aspektů získáváme o sítích mnoho dalších informací. Přidáme-li k sítím časový rozměr, např. dobu aktivity hran nebo vrcholů, získáme dynamickou tzv. temporální síť.
Temporální sítě projektují údaje o tom, kdy se věci dějí, z dynamických systémů do sítí, čili do základních struktur, nad kterými se dynamika odehrává. Výzkum v oblasti statických sítí je mnohem širší než v oblasti sítí temporálních, nicméně právě oblast temporálních sítí se prudce rozvíjí. Tok informací prostřednictvím e-mailových zpráv nebo např. sociální média jsou příkladem systémů, které se v poslední době těší velké pozornosti.
1. Seznamte se s komplexními sítěmi, s jejich topologickými vlastnostmi, které se nejčastěji zkoumají a s algoritmy pro jejich analýzu.
2. Seznamte se s temporálními sítěmi, s jejich typy, s jejich reprezentací a s modifikacemi algoritmů pro analýzu sítí zohledňující jejich časový rozměr.
3. Vyberte a naimplementujte algoritmy pro analýzu temporálních sítí.
4. Nad vhodnými datovými kolekcemi proveďte experimenty, experimenty vyhodnoťte a jejich výsledky vhodně reprezentujte.

Seznam doporučené odborné literatury:

[1] Petter Holme and Jari Saramäki, Temporal networks, Phys. Rep. 519, 97-125 (2012)
https://arxiv.org/pdf/1108.1780v2.pdf
[2] Petter Holme, Modern temporal network theory: A colloquium, Eur. Phys. J. B 88, 234 (2015),
https://arxiv.org/pdf/1508.01303v3.pdf
[3] Podle pokynů vedoucího diplomové práce

Formální náležitosti a rozsah diplomové práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí diplomové práce: **RNDr. Eliška Ochodková, Ph.D.**

Datum zadání: 01.09.2016

Datum odevzdání: 28.04.2017

doc. Dr. Ing. Eduard Sojka
vedoucí katedry

prof. RNDr. Václav Snášel, CSc.
děkan fakulty

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Ostravě 28. 4. 2017 Podpis .....

Souhlasím se zveřejněním této diplomové práce dle požadavků čl. 26, odst. 9 Studijního a zkušebního řádu pro studium v magisterských programech VŠB-TU Ostrava.

V Ostravě 28. 4. 2017 .....

Rád bych na tomto místě poděkoval vedoucí této práce RNDr. Elišce Ochodkové, Ph.D., za velkou pomoc při jejím zpracování.

Abstrakt

Tato diplomová práce se zaměřuje na analýzu temporálních sítí. V první části se věnuje stručnému úvodu do temporálních sítí a do teorie grafů, kde jsou popsány základní pojmy a vlastnosti z teorie grafů. Tyto vlastnosti pak byly analyzovány na vybraných datových kolekcích pomocí vlastního nástroje, nebo pomocí nástroje R.

Klíčová slova: Java, Graf, Temporální Síť, Uzel, Hrana, Centralita

Abstract

This theses focuses on temporal network analysis. At the beginning is a brief introduction to temporal networks and graph theory, where the basic concepts and properties of graph theory are described. These properties were then analyzed on selected data collections using the own tool or the R tool.

Key Words: Java, Graph, Temporal Network, Node, Edge, Centrality

Obsah

Seznam použitých zkratek a symbolů

TS	–	Temporální Síť
ATS	–	Analýza Temporálních Sítí
DFS	–	Depth-first search
DBLP	–	Digital Bibliography & Library Project

Seznam obrázků

Seznam tabulek

1 Úvod

Velké množství systémů v přírodě, společnosti a technologiích - od sociálních sítí na internetu, přes rozvodné sítě až po nervové soustavy - může být modelováno jako grafy, množinami uzlů spojených hranami. Struktura sítě nám pomáhá pochopit, předpovídat a optimalizovat chování dynamického systému. Avšak v mnoha případech nemusí být hrany i uzly stále aktivní. Příkladem může být SMS, nebo emailová komunikace, nebo telefonní hovory, kde hrany představují aktuální kontakt.

Cílem této práce je zaměřit se na vybrané datové kolekce temporálních sítí a provést nad nimi experimenty. Cílem je taky implementovat vlastní aplikaci, pomocí které provedeme analýzu zvolených datových kolekcí.

V první kapitole si povíme něco o temporálních sítích a představíme si některé příklady temporálních sítí. V další kapitole si stručně projdeme základní teorii grafů, od statických grafů pomalu přejdeme k temporálním grafům a jak je můžeme reprezentovat. Ve třetí kapitole si popíšeme základní vlastnosti temporálních sítí, které budeme zkoumat (stupeň, nejkratší cesta, centralita, atd.). V páté kapitole se zaměříme na popis a návrh aplikace. Představíme si softwarovou reprezentaci datových kolekcí představující temporální sítě a implementované algoritmy. V šesté kapitole se zaměříme na samotnou analýzu. Popíšeme si datové kolekce, jednotlivé experimenty, cíle a taky výsledky těchto experimentů. Na závěr zhodnotíme výsledky. Podíváme se taky na některé problémy které se vyskytly během implementace, nebo během experimentů a bylo třeba je vyřešit.

2 Temporální sítě

Jak už bylo naznačeno v úvodu, velké množství reálných systémů okolo nás se dá chápat jako nějaký systém, který se dá reprezentovat sítí, tedy kolekcí entit spojených vazbou a tyto sítě pak můžeme reprezentovat grafem. Graf, který je množinou uzlů a hran, je ale pouze nejjednodušším typem sítě. Ve skutečnosti bývají sítě mnohem komplexnější, v sítí mohou být uzly a hrany více druhů, mohou mít různé parametry. Jedná-li se například o sociální síť, uzly mohou reprezentovat muže a ženy a obsahovat mnoho informací o nich jako je jméno, věk atd. Hrany mohou reprezentovat typ spojení například přátelé, příbuzní, nebo kolegové z práce a taky mohou obsahovat informace o tom jak dlouho nebo jak dobře se tito dva lidé znají.

V této práci se ale takto do hloubky sítím věnovat nebudeme a k analýze nám postačí pouze základní reprezentace grafu a informace o čase, v jakém byly hrany mezi uzly aktivní. V této kapitole se podíváme na některé příklady temporálních sítí a něco si o nich řekneme. Informace byly čerpány ze zdroje [2].

2.1 Sítě one-to-one

Záznamy komunikace mezi lidmi jsou typickým příkladem temporální sítě. Komunikace je většinou ve formě zpráv které přicházejí v určitých časových okamžicích, nebo to muže být dialog mezi dvěma osobami v rámci časového intervalu. Typicky to může být emailová komunikace, SMS komunikace, nebo telefonní hovory. Analýza centralit (viz kap. 4.3) takovýchto sítí pak může být užitečná při navrhování strategií proti šíření škodlivého softwaru mezi mobilními zařízeními.

2.2 Sítě one-to-many

Jedná se o sítě, kde se informace šíří od jednoho zdroje k více příjemcům. Příkladem může být hromadný email, který jedna osoba rozešle v daný časový okamžik všem, nebo jen některým, uživatelům v síti. Dalším příkladem může být člověk, který přednáší a po nějaký časový interval ho poslouchá skupina lidí. V počítačové síti se takovému šíření zprav říká broadcast, jeden počítač zprávy vysílá a ostatní je přijímají.

2.3 Síť kontaktů

Tyto temporální sítě jsou sítě fyzických kontaktů mezi lidmi, například na pracovišti, nebo ve škole. Analýza těchto sítí byla dlouhou dobu dosti náročná, avšak v dnešní době je to mnohem snadnější díky moderním elektronickým zařízením. Průkopníci zde byli studenti z Massachusetts Institute of Technology, kteří byli schopni pomocí bluetooth v mobilních telefonech analyzovat vzdálenosti mezi sebou.

2.4 Biologické síť

V mikrobiologii existuje mnoho systému, které mohou být modelovány jako sítě, ale ne všechny jsou natolik dynamické, aby se daly modelovat jako temporální sítě. Jedním ze systémů, které se jako temporální síť chovají, jsou molekulární interakce v buňce, nazývané interactome. Uzly jsou zde proteiny a jiné molekuly, které se navzájem spojují a vytvářejí tak biologické funkce. Nejčastěji jsou tyto interakce reprezentovány jako statická síť, nicméně se dá vycházet ze skutečnosti, že spojení nejsou aktivní po celou dobu.

Další biologickou sítí, která se mění v průběhu času je metabolismus, množina chemických reakcí v organismu. Uzly v metabolické síti jsou druhy molekul, které jsou spojeny, pokud jsou zapojeny do stejné chemické reakce a v daném okamžiku je aktivní pouze část biochemického reakčního systému.

2.5 Neuronové sítě

Síť neuronů je dalším příkladem temporální sítě, která patří do biologických sítí, a kterou je možné modelovat jako temporální síť. Existuje několik úrovní temporálních spojení, od základních spojení mezi jednotlivými neurony, po fyziologické nebo funkční spojení mezi části mozku. Ve druhém případě se používají různé metody měření, například Elektroencefalografie (EEG) a Magnetoencefalografie (MEG) měřící elektrické signály a odchylky jednotlivých extrakraniálních magnetických polí. Tyto metody jsou poměrně přesné z časového hlediska, ale mají méně přesné prostorové rozlišovací schopností. Funkční magnetická rezonance (fMRI) měří mozkovou aktivitu na základě hladiny okysličení krve, což má sice přesnější prostorové rozlišení, ale naopak horší časové rozlišení. K výsledkům je tedy nutné přistupovat s určitou rezervou.

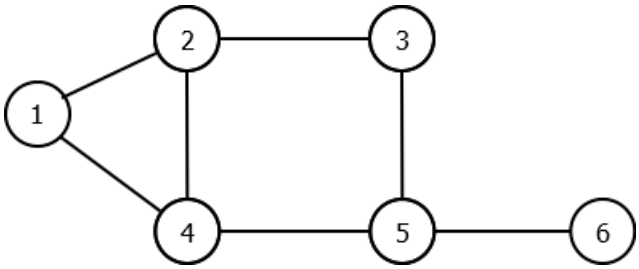
Výše uvedené příklady nejsou ani zdaleka všechny příklady temporálních sítí. Stačí se podívat na jakoukoliv síť a zamyslet se, jestli se v průběhu času nějak mění, a jestli má smysl o takovéto síti jako o temporální uvažovat.

3 Teorie grafů

Teorie grafů je obor diskrétní matematiky, který zkoumá vlastnosti grafů, tedy množinami uzlů a hran, kde každá hrana je určena dvěma uzly a volitelně směrem a, nebo váhou („cenou“). Váha může znamenat například délku cesty, náklady na přesun nebo průchodnost. V této kapitole si nejprve nadefinuje pojem graf a podíváme se na různé typy grafů a jejich odlišnosti. Dále si nadefinujeme základní pojmy z teorie grafů, kterými se budeme dále v této práci zabývat, a také si ukážeme, jak můžeme grafy reprezentovat.

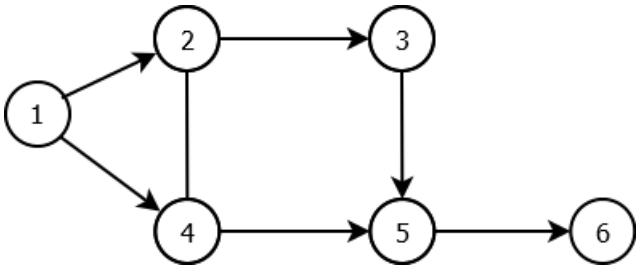
3.1 Graf

Definice grafu podle [1] říká, že graf G (také jednoduchý graf) je uspořádaná dvojice $G = (V, E)$, kde V je neprázdná množina uzlů a E je množina hran - množina (některých) dvouprvkových podmnožin množiny V. Příklad jednoduchého grafu je na následujícím obrázku 1.

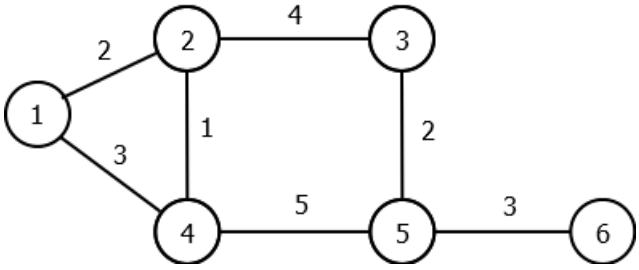


Obrázek 1: Jednoduchý graf

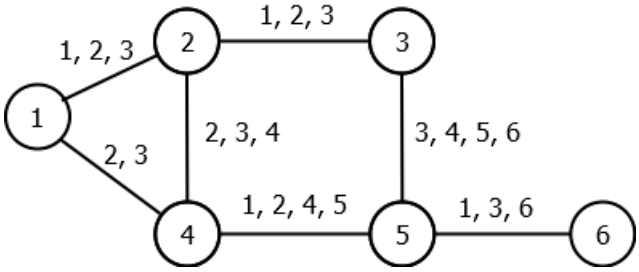
Takto nadefinovaný graf je graf neorientovaný, protože nemůžeme říct ze kterého a do kterého uzlu hrana vede. Přidáním šipek tuto orientaci určíme a vnikne tak orientovaný graf, příklad je na obrázku 2. A pokud jednotlivým hranám přidáme ještě hodnotu, dostaneme ohodnocený graf, obrázek 3. U neohodnoceného grafu má každá hrana hodnotu 1. V této práci se orientovaným a ohodnoceným grafům věnovat nebudeme, místo toho přidáme hranám časové okamžiky, nebo intervaly a dostaneme tím dynamický graf, nebo-li temporální síť, obrázek 4.



Obrázek 2: Orientovaný graf



Obrázek 3: Ohodnocený graf



Obrázek 4: Temporální graf

3.2 Základní pojmy z teorie grafů

Stupeň uzlu

V teorii grafů se pojmem stupeň uzlu označuje počet uzlů, které jsou s daným uzlem přímo spojené hranou. Pokud má hrana nějaký uzel jako koncový říkáme, že hrana s tímto uzlem inciduje. Stupeň uzlu u se značí *deg(u)*. U orientovaného grafu se ještě rozlišuje zda hrana do uzlu vstupuje, nebo z něho vystupuje. Budeme mít tedy vstupní a výstupní stupeň uzlu a značíme je $deg_{in}(u)$ a $deg_{out}(u)$.

Cesta

Cesta je posloupnost uzlů, pro kterou platí, že v grafu existuje hrana z daného uzlu do následujícího uzlu. Žádné dva uzly (a tedy ani hrany) se přitom neopakují. Podle [1] je definice cesty následující: Graf na n uzlech, které jsou spojeny po řadě $n-1$ hranami se nazývá cesta.

U temporálních sítí se používá pojem čas respektující cesta [2], při průchodu touto cestou je nutné respektovat čas ve kterém jsou hrany aktivní. Jedná se tedy o množinu uzlů spojených hranami s neklesajícím časem. Na rozdíl od jednoduchých sítí tady například nemusí platit tranzitivita. Pokud je hrana (u_0, u_1) aktivní pouze v pozdějším čase než hrana (u_1, u_2), potom není možné se z uzlu (u_0) do uzlu (u_2) dostat.

Vzdálenost

Vzdálenost mezi uzly u a v značíme *d(u, v)* a je dána délkou nejkratší cesty mezi uzly u a v v Grafu G. Pokud cesta mezi uzly u a v neexistuje, vzdálenost *d(u, v)* se bude rovnat nekonečnu.

Latence

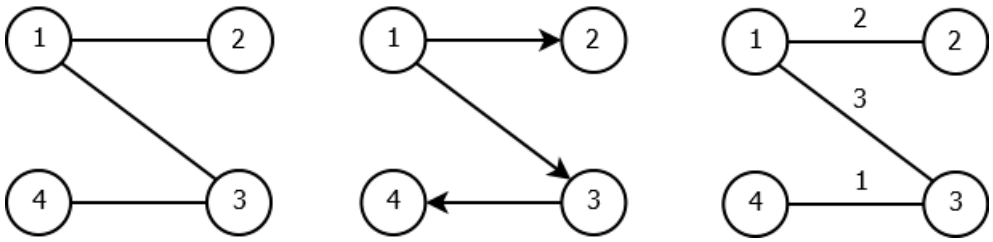
Latence je nejkratší doba trvání čas respektující cesty mezi dvěma uzly v jednotkách, které si určíme.

Počet uzlů a hran

Dále v tomto textu budeme využívat pojmy počet uzlů a počet hran, tak bychom si mohli ukázat jak se značí. Počet uzlů značíme $|V|$, nebo malým písmenkem n. Počet hran označujeme $|E|$, nebo také m.

3.3 Reprezentace grafu

K tomu, abychom mohli síť softwarově analyzovat, je třeba jí vhodně reprezentovat. K tomu slouží několik datových struktur, které si teď popíšeme. Jako příklad nám poslouží následující grafy na obrázku 5.



Obrázek 5: Jednoduchý, orientovaný a ohodnocený graf

Matice incidence

První datovou strukturou pro reprezentaci grafu je incideční matice, značíme I. Velikost matice je $n \times m$, kde n je počet uzlů, a m je počet hran. Je-li daná hrana spojena s daným uzlem bude v matici hodnota 1, pokud ne bude hodnota 0. Orientované grafy dále vyžívají hodnotu -1 k rozlišení zda hrana do uzlu vstupuje nebo z něho vystupuje. Pokud tedy budeme mít orientovanou hranu z uzlu u do uzlu v, bude v matici na pozici u hodnota 1 a na pozici v hodnota -1.

$$I = \begin{pmatrix} 1 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{pmatrix}, I_{orient} = \begin{pmatrix} 1 & -1 & 0 \\ -1 & 0 & 0 \\ 0 & 1 & 1 \\ 0 & 0 & -1 \end{pmatrix}$$

Matice sousednosti

Další možností reprezentace grafu je matice sousednosti, značíme A. Velikost matice je $n \times n$. Jsou-li dva uzly spojeny hranou bude hodnota v matici 1, jinak 0. Matice pro neorientovaný graf

bude symetrická, pro orientovaný graf nikoli, protože hodnota 1 bude pouze u uzlů do kterých hrana vstupuje.

$$A = \begin{pmatrix} 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}, A_{orient} = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{pmatrix}, A_{ohod} = \begin{pmatrix} 0 & 2 & 3 & 0 \\ 2 & 0 & 0 & 0 \\ 3 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}$$

Seznam sousedů

Nevýhodou výše popsaných matic je, že ve většině případů se jedná o řídké matice, uzly jsou spojeny jen s několika dalšími uzly. Hodí se tak spíše pro menší grafy. Pro větší grafy se proto lépe hodí seznam sousedů, kde pro každý uzel máme seznam jeho sousedů.

Uzel	Seznam sousedů
1	2, 3
2	1
3	1, 4
4	3

Tabulka 1: Seznam sousedů

Seznam hran

Obdobou seznamu sousedů je seznam hran, kde máme seznam dvojic uzlů, představujících hranu. Seznam hran je ideální i pro reprezentaci ohodnoceného, nebo temporálního grafu, kde pro každou hranu můžeme snadno definovat váhu, časové okamžiky, nebo intervaly, případně i jiné vlastnosti.

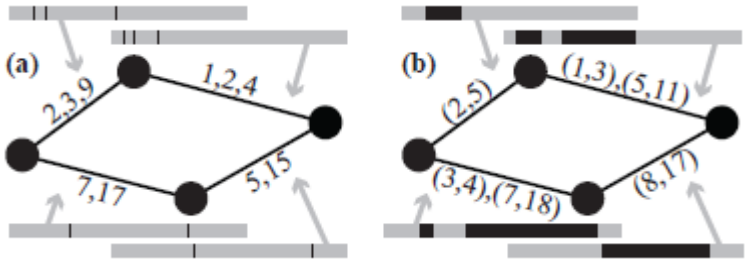
Uzel 1	Uzel 2	Váha
1	2	2
1	3	3
3	4	1

Tabulka 2: Seznam hran

V tabulkách 3 a 4 jsou příklady reprezentace temporálních grafů z obrázku 6 pomocí seznamu hran a časových okamžiků, respektive časových intervalů, ve kterých jsou hrany aktivní.

Uzel 1	Uzel 2	Časové okamžiky
1	2	2, 3, 9
2	3	1, 2, 4
3	4	5, 15
4	1	7, 17

Tabulka 3: Seznam hran s časovými okamžiky



Obrázek 6: Temporální grafy [2]

Uzel 1	Uzel 2	Časové intervaly
1	2	(2, 5)
2	3	(1, 3), (5, 11)
3	4	(8, 17)
1	4	(3, 4), (7, 8)

Tabulka 4: Seznam hran s časovými intervaly

4 Vlastnosti sítí

V této kapitole si popíšeme vlastnosti sítí, které budeme v této práci analyzovat. Vlastnosti rozdělujeme na dva typy, lokální, které se vztahují ke konkrétnímu uzlu a globální vlastnosti pro celou síť.

4.1 Lokální vlastnosti

Distribuce stupňů

Distribuce stupnů je pravděpodobnost, s jakou bude daný uzel stupně. Spočteme ji jako podíl počtu uzlů daného stupně ke všem uzlům.

Shlukovací koeficient

Shlukovací koeficient nám udává, jaká je pravděpodobnost, že dva libovolné uzly, které mají alespoň jednoho společného souseda, jsou také spojeny hranou. Vzorec pro výpočet je:

$$C_u = \frac{2e_u}{deg(u)(deg(u) - 1)} \tag{1}$$

kde e je počet navzájem propojených sousedů uzlu u.

Pro temporální síť je nutné, při výpočtu shlukovacího koeficientu, kontrolovat aktivitu hran. V tomto případě je shlukovací koeficient pravděpodobnost, že dva libovolné uzly, které mají alespoň jednoho společného souseda, spojeného hranou ve stejný časový okamžik, jsou také spojeny hranou ve stejný časový okamžik.

$$C_u = \frac{2e_u^t}{deg(u)(deg(u) - 1)} \tag{2}$$

v tomto případě e je počet navzájem propojených sousedů uzlu u v čase t [3].

Centralita

Centralita obecně určuje, jak je uzel důležitý z hlediska struktury sítě. Každá centralita je založena na určitých předpokladech, a proto je důležité zvážit, zda tyto předpoklady zapadají do naší oblasti zájmu. V této práci se budeme věnovat closeness a betweeness centralitám, obě jsou založeny na nejkratších cestách.

Centrality založené na cestách nejsou metriky, nesplňují například axiom symetrie, nebo trojúhelníkovou nerovnost. Často obdržíme jiné výsledky než metrikami, které určují důležitost vzhledem ke stupňům uzlu.

Closeness cetralita

Closeness centralita udává, jak je daný uzel schopen komunikovat s ostatními uzly. Vzorec pro výpočet je:

$$C_u = \frac{1}{n-1} \sum_{v \neq u} d_{u,v} \tag{3}$$

kde $d_{u,v}$ je vzdálenost mezi uzly u a v. Je dána jako průměr nejkratších cest z daného uzlu do všech ostatních uzlů. Počítá se vždy v rámci jedné komponenty souvislosti. Porovnání hodnot mezi komponentami nemá správnou vypovídající hodnotu. Proto se někdy používá následující vzorec:

$$C_u = \frac{1}{n-1} \sum_{v \neq u} \frac{1}{d_{u,v}} \tag{4}$$

U temporálních sítí se closeness centralita počítá vzhledem k latenci a označovat ji budeme jako temporální closeness centralita. Vzorec vypadá následovně:

$$C_{(u,t)} = \frac{N}{\sum_{v \neq u} \lambda_{u,t}(v)} \tag{5}$$

kde N je počet uzlů a $\lambda_{u,t}(v)$ je latence čas respektující cesty mezi uzly u a v.

Betweeness centralita

Tato centralita nám udává důležitost uzlu podle toho kolik nejkratších cest daným uzlem prochází. Čím více nejkratších cest uzlem prochází, tím je uzel důležitější. Pro výpočet betweeness centrality se využívá následující vzorec:

$$C_u = \sum_{v \neq u, w \neq u} \frac{p_{v,w}(u)}{p_{v,w}} \tag{6}$$

kde $p_{v,w}$ znamená počet cest z uzlu v do uzlu w a $p_{v,w}(u)$ je počet cest z uzlu v do uzlu w procházejících uzlem u. Pro temporální sítě se počítají cesty respektující čas a budeme ji říkat temporální betweenness centralita.

4.2 Globální vlastnosti

Průměr a průměrná délka nejkratší cesty

V předchozí kapitole jsme si nadefinovali pojem vzdálenost. Průměrem grafu je nejdelší vzdálenost mezi libovolnými uzly. Průměr je nejdelší nejkratší cesta, je to největší hodnota v matici délek nejkratších cest.

Průměrná délka nejkratší cesty je dána průměrem vzdáleností nejkratších cest mezi všemi dvojicemi uzlů. Vypočteme ji pomocí následujícího vzorce:

$$d_{avg} = \frac{\sum_v^n \sum_{v \neq u}^n d(u,v)}{n(n-1)} \tag{7}$$

V temporálních grafech se průměr a průměrná délka nejkratší cesty počítá vzhledem k latenci.

Komponenta souvislosti

Komponenta souvislosti je maximální souvislý podgraf, t.j. v tomto podgrafu najdeme cestu z uzlů u do uzlů v pro jakékoliv uzly u a v v podgrafu. Souvislý graf má právě jednu komponentu souvislosti.

Hustota

Hustý graf je takový graf, kde počet hran se blíží maximálnímu počtu hran. Naopak graf s jen několika hranami je řídký graf. Rozdíl mezi hustým a řídkým grafem není přesně definován a záleží vždy na kontextu. Pro neorientovaný graf a orientovaný graf jsou hustoty definovány jako:

$$D_{orient} = \frac{2|E|}{|V|(|V|-1)}, D_{neorient} = \frac{|E|}{|V|(|V|-1)} \tag{8}$$

kde E je množina hran a V je množina uzlů.

Modularita

Modularita nám určuje rozdělení grafu do modulů, nazývaných také komunity. Sítě s vysokou modularitou mají husté spojení mezi uzly v rámci komunit, ale řídké spojení mezi uzly z různých komunit. Modularita je často používána v algoritmech pro detekci komunitní struktury v sítích. Nevýhodou této metody je, že není vhodná pro hledání malých komunit. Modularita Q se spočte podle vzorce:

$$Q = \frac{1}{2m} \sum_{i,j \in V} [A_{i,j} - \frac{k_i k_j}{2m}] \delta(c_i, c_j) \tag{9}$$

kde m je počet hran sítě, $A_{i,j}$ je váha hrany mezi uzly i a j, k_i a k_j jsou stupně těchto uzlů a $\delta(c_i, c_j)$ je funkce která vrací 1 pokud jsou uzly ve stejné komunitě a 0 pokud se nacházejí v různých komunitách [4].

Asortativita

Vyjadřuje tendenci uzlů spojovat se s uzly, které jsou v něčem podobné, v teorii grafu to je obvykle stupeň uzlu. Asortativita se často určuje jako korelace mezi uzly. Existuje několik způsobu

jak tuto korelaci spočítat. Dvě nejvýznamnější metody jsou koeficient asortativity a konektivita sousedů [5].

Odhad mocninného exponentu

U velkých sítí obvykle bývá mnoho uzlů s velmi malým stupněm a jen málo uzlů s velkým stupněm. Graf distribuce stupňů potom odpovídá mocninnému rozdělení. Nalezením mocninného exponentu pak můžeme spočítat pravděpodobnost s jakou bude uzel daného stupně. Tento vztah vypadá následovně:

$$p(d) \approx d^{-\alpha} \tag{10}$$

kde d je stupeň a α je mocninný exponent.

5 Návrh aplikace

V této kapitole se blíže podíváme na implementaci aplikace. Aplikace je napsaná v programovacím jazyce Java. Ten jsem zvolil proto, že mi je ze všech programovacích jazyků nejbližší, je objektově orientovaný a umožňuje pohodlnou práci s datovými kolekcemi typu seznam, množina, nebo slovník. Díky objektově orientovanému přístupu je potom snadné pracovat s uzly a hranami jako s objekty.

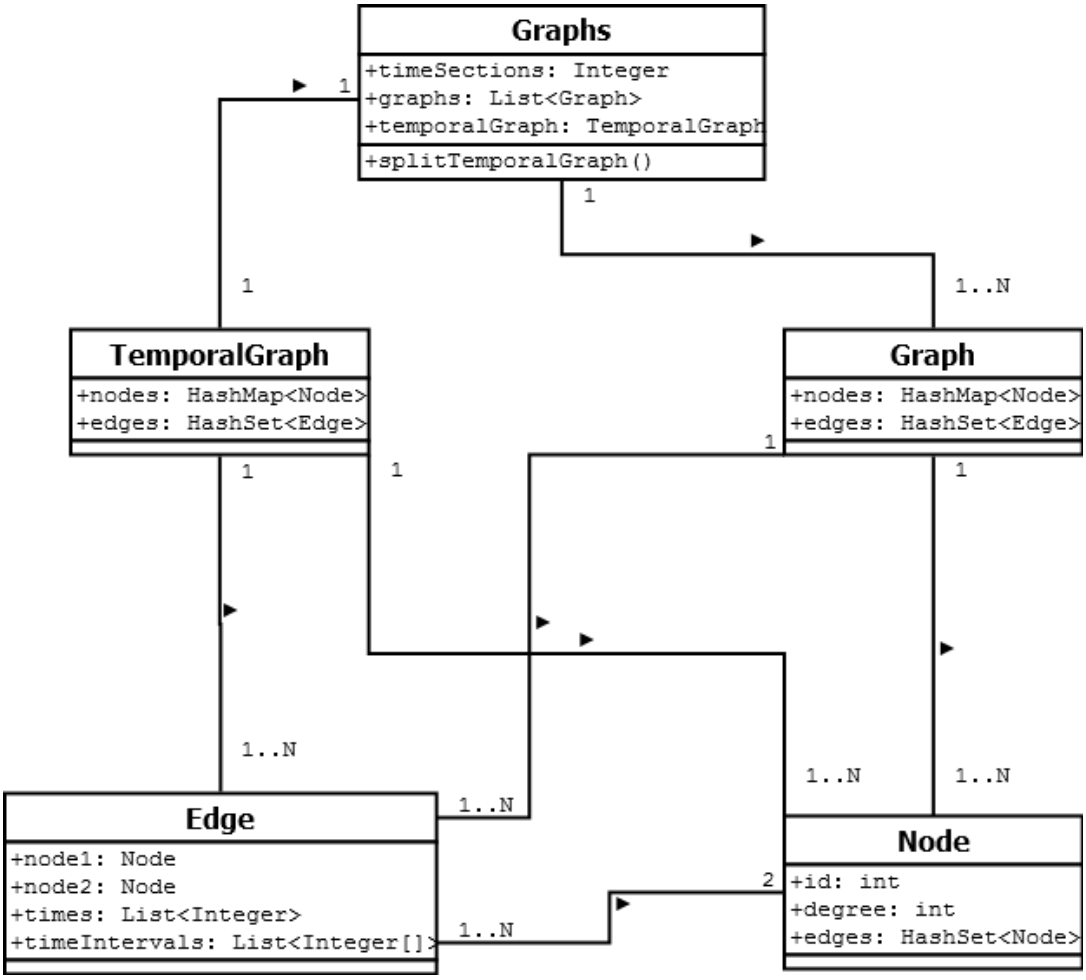
5.1 Architektura aplikace

Páteří celé aplikace jsou třídy *Node* a *Edge* reprezentující uzly a hrany grafu. Třída *Edge* má dvě instanční proměnné typu *Node* ve kterých jsou uloženy koncové uzly hrany. Dále pak má dvě proměnné typu *ArrayList<Integer>* pro seznam časů a časových intervalů ve kterých je hrana aktivní. Třída *Node* má dvě proměnné typu int kde je uloženo *id* uzlu a *číslo* uzlu. Rozdíl je v tom, že číslo uzlu je načteno ze vstupního souboru a může to být libovolné celé číslo, ale *id* vždy začíná od nuly a s každým novým uzlem se zvyšuje o jedničku a slouží pro práci s maticemi. Další dvě proměnné které třída *Node* obsahuje jsou proměnné typu *HashSet<Node>* a *HashSet<Edge>*, nebo-li množina uzlů a hran a představující všechny sousedy a hrany daného uzlu. Dále obsahuje několik proměnných typu *int* pro ukládáni stupně a *double* pro ukládání closeness centrality, betweness centrality a shluovacího koeficientu. Pro reprezentaci sítě a temporální sítě slouží třídy *Graph* a *TemporalGraph*. Základem těchto tříd jsou kolekce obsahující seznam všech uzlů a hran a dále metody pro analýzu. Dále tyto třídy obsahují několik pomocných kolekcí pro výsledky výpočtů. Hlavní rozdíl mezi těmito třídami je, jak už z názvu vyplývá, že třída *TemporalGraph* pracuje s temporální sítí. S těmito dvěmi třídami pak pracuje třída *Graphs*, která rozdělí čas na několik časových oken a z jedné temporální sítě, tak udělá několik jednoduchých sítí, které uloží do proměnné typu *ArrayList<Graph>*. Jaký je mezi těmito třídami vztah můžeme vidět na následujícím obrázku 7, jednotlivé třídy kvůli přehlednosti neobsahují všechny proměnné a metody.

Dále je implementováno několik pomocných tříd, které slouží pro načtení datových kolekcí ze vstupního souboru, zapsání výsledků do souboru a načtení konfiguračního souboru. Konfigurační soubor slouží pro nastavení vstupních parametrů aplikace.

K načtení konfiguračního souboru slouží třída *Config*, která za souboru *config.cfg* načte vstupní informace, například vstupní soubor, znak oddělující jednotlivé uzly ve vstupním souboru a v neposlední řade počet časových oken, na která chceme rozdělit temporální síť. Pokud temporální síť nechceme rozdělit na časová okna, zadá se nula.

Pro vytvoření sítě ze vstupního souboru slouží třída *Reader*, která obsahuje tři statické metody. Metoda *readData* slouží k načtení dat ze souboru. Má dva parametry typu *String*, jméno souboru a znak oddělující data, nejčastěji jsou to znaky čárka ","nebo středník ";". Vstupní soubor je textový soubor obsahující seznam hran. Na každém řádku je dvojice uzlů představujících hranu, v případě temporálních sítí je za touto dvojicí uzlů i časový údaj. Všechny hodnoty jsou



Obrázek 7: Diagram tříd

oddělené určitým znakem. Metoda vrací seznam polí, kde každé pole představuje jeden řádek a každá položka v poli jednu hodnotu ze vstupního souboru oddělenou příslušným znakem. Tuto metodu využívají další dvě metody, které z načtených dat vygenerují statickou nebo temporální síť.

Třída *Utils* slouží pro zápis dat do souborů. Jedná se o seznamy uzlů s údaji o stupních, centralitách a shlukovacím koeficiendu. Dále to jsou pak distribuce stupňů, distribude velikosti komponent souvislosti a distribuce shlukovacího koeficientu.

5.2 Algoritmy

Většina výpočtů byla poměrně snadných. Některé výpočty ale vyžadovaly vyřešit složitější problémy, například hledání komponent souvislosti, hledání nejkratších cest respektujících čas, nebo výpočet betweenness centrality, který byl obzvláště časově náročný.

Hledání komponent souvislosti

Pro hledání počtu komponent souvislosti použijeme lehce upravený rekurzivní algoritmus prohledávání do hloubky (DFS). Nejprve si označíme všechny uzly jako nenavštívené (FRESH) a uložíme si je do pole. Pak zavoláme rekurzivní metodu DFS z prvního uzlu, která postupně projde všechny uzly dosažitelné z prvního uzlu. Zpracované uzly pak označíme jako zavřené (CLOSED). Pokud se v poli již nenacházejí žádné další uzly, síť má právě jednu komponentu souvislosti a algoritmus končí. V opačném případě se zavolá metoda DFS na dalším uzlu ve stavu FRESH a počet nalezených komponent se zvýší o 1. Algoritmus navíc ještě počítá počet uzlů které jednotlivé komponenty obsahují a ukládá si je pro výpis distribuce komponent souvislosti.

```
private static final int FRESH = 0; // Dosud nezpracovany uzel
private static final int OPENED = 1; // Zpracovavany uzel
private static final int CLOSED = 2; // Jiz zpracovany uzel
public static int countComponents(){
   int[] state = new int[nodes.size()]; // Stavy jednotlivych uzlu
   for(int i = 0; i < state.length; i++) {
      state[i] = FRESH;
   }
   int counter = 0; // Pocitadlo komponent
   int componentSize; // Pocitadlo uzlu v komponente
   //zajisti pruchod vsemi komponentami souvislosti
   for(int i = 0; i < nodes.size(); i++){
      if(state[i] == FRESH){
         counter++;
         componentSize = doDFS(graph, i, state);
         // Ukladani velikosti komponent souvislosti
         if (this.componentsDistribution.containsKey(componentSize)) {
            this.componentsDistribution.put(componentSize, this.
               componentsDistribution.get(componentSize) + 1);
         } else {
            this.componentsDistribution.put(componentSize, 1);
         }
      }
   }
   return counter;
}
/**
 * Rekurzivni prohledavani do hloubky
 * @param nodeId cislo uzlu
```

```
 * @param state stavy uzlu
 */
private static void doDFS(int nodeId, int[] state) {
   state[nodeId] = OPENED;
   int counter = 1;
   for(Node i : this.nodes.get(nodeId).getNeighbors()){
      if(state[i.getId()] == FRESH) {
         this.biggestComp.add(i);
         counter += doDFS(i.getId(), state);
      }
   }
   state[nodeId] = CLOSED;
   return counter;
}
```

Výpis 1: Hledání komponent souvislosti

Výpočet nejkratších cest respektujících čas

Pro výpočet nejkratších cest respektujících čas použijeme upravený algoritmus prohledávání do šířky. Algoritmus prochází všechny časové okamžiky. Na začátku máme jeden uzel ze kterého vycházíme. Tento uzel přidáme do množiny dosažených uzlů a prohledáváme jeho okolí, kam se můžeme v daný okamžik dostat. Uzly, kam jsme se dostali opět přidáme do množiny dosažených uzlů a zároveň si do matice uložíme čas ve který jsme se do uzlu dostali, pokud tam již není uložený kratší čas. S dalším časovým okamžikem opět prohledáváme okolí všech těchto dosažených uzlů.

Nevýhodou je, že si algoritmus nepamatuje cestu kudy se do uzlů dostal, proto se nehodí například pro výpočet betweenness centrality, je ale podstatně rychlejší než prohledávání do hloubky, které je použito pro výpočet betweenness centrality.

```
public int[][] countShortestPathsRespectingTime() {
   int m = this.nodes.size();
   int[][] matrix = new int[m][m];
   Set<Node> dosazene = new HashSet<>();
   Set<Node> dosazitelne = new HashSet<>();
   for (Node node : this.nodes) {
      // Vymazu pomocne seznamy uzlu
      dosazene.clear();
      dosazitelne.clear();
      for (int t = 0; t < this.getLastTime(); t++) {
         dosazene.add(node); // Pridam prvni uzel
```

```
            dosazitelne.addAll(node.getNeighbors()); // Pridam jeho sousedy
            Set<Node> noveDosazitelne = new HashSet<>();
            Set<Node> noveDosazene = new HashSet<>();
            // Pro vsechny dosazene uzly prohledavam jeho sousedy
            for (Node n1 : dosazene) {
               for (Node n2 : n1.getNeighbors()) {
                  if (isActive(n1, n2)) {
                     if (matrix[node.getId()][n2.getId()] > t) {
                        matrix[node.getId()][n2.getId()] = t;
                        noveDosazene.add(n2);
                        noveDosazitelne.addAll(n2.getNeighbors());
                     }
                  }
               }
            }
            dosazene.addAll(noveDosazene);
            dosazitelne.addAll(noveDosazitelne);
            noveDosazitelne.clear();
         }
      }
      return matrix;
   }
```

Výpis 2: Výpočet nejkratších cest respektujících čas

Výpočet temporální betweenness centrality

Pro výpočet betweenness centrality je nutné najít všechny nejkratší cesty respektující čas, které uzlem procházejí. V případě, že jich je více je nutné si zapamatovat všechny. Cest v síti mezi dvěma uzly může existovat velké množství, tady to navíc ještě komplikuje čas, ve kterém jsou hrany aktivní. Tedy i v případě, že jsou dva uzly sousedé, je nutné hledat, zda není možné se do daného uzlu dostat dříve, byť za cenu průchodu více uzly.

Algoritmus je bohužel časově velice náročný a určitě je zde velký prostor pro zlepšení. Zvolil jsem prohledávání do hloubky, které ale nebylo tak jednoduché jako v případě hledání komponent souvislosti. Výpis kódu rekurzivní metody provádějící hledání je uveden níže.

```
/**
 * Rekurzivni prohledavani do hloubky
 * @param visited navstivene uzly
 * @param root vychozi uzel
 * @param dest cilovy uzel
```

```
 * @param node aktualni uzel
 * @param time aktualni cas
 */
public int doDFS(Stack<Node> visited, Node root, Node dest, Node node, int time
    ) {
   int count = 0;
   visited.push(node);
   for (Edge e: node.getEdges()) {
      Node v = e.getSecondNode();
      int totalTime = e.getNearestTime(time);
      if (totalTime == -1) {
         continue;
      }
      if (this.shortestTime >= totalTime && v.equals(dest)) {
         Stack<Node> newShortestPath = new Stack<>();
         newShortestPath.addAll(visited);
         if (this.shortestTime > totalTime) {
            this.shortestPaths.clear();
            this.shortestTime = totalTime;
         }
         this.shortestPaths.add(newShortestPath);
         count++;
      } else if (this.shortestTime >= totalTime && !visited.contains(v) &&
          totalTime <= dest.getLastTime()) {
      count += this.doDFS(visited, root, dest, v, totalTime + 1);
      }
   }
   visited.pop();
   return count;
}
```

Výpis 3: Hledání nejkratší cesty respektující čas mezi dvěma uzly

Nalezené cesty se ukládají do seznamu cest, ten se na konci projde a pro všechny uzly, v něm uložené, se navýší počítadlo cest procházejících uzlem a vydělí se počtem všech nejkratších cest.

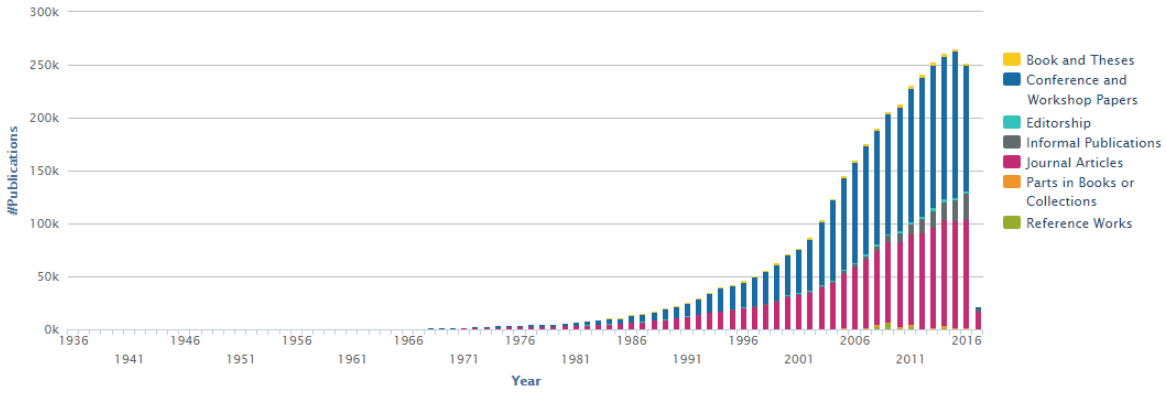
6 Experimenty

V této kapitole se podíváme na čtyři různé datové kolekce temporálních sítí a provedeme nad nimi experimenty, jejichž cílem bude analyzovat vlastnosti, které byly popsané v předchozí kapitole. Během experimentu se na síť budeme dívat dvěma způsoby, analyzujeme temporální síť jako celek a jako sekvenci časových oken. Výjimkou bude síť DBLP, kde budeme analyzovat pouze časová okna.

Při analýze použijeme vlastní aplikaci i nástroj R [6]. Nabízel se i nástroj Gephi[1], ale nedokázal zpracovat velké sítě. Pomocí nástroje R spočítáme modularitu, asortativitu a betweenness centralitu, zároveň nám R posloužil i pro kontrolu výsledků získaných vlastní aplikací.

6.1 DBLP

DBLP computer science bibliography (Digital Bibliography & Library Project) je webová služba otevřeně poskytující bibliografické údaje o hlavních vědeckých časopisech a sbornících. Tento projekt začal v roce 1993. Na obrázku 8 je vidět graf počtu publikací od roku 1936 až po rok 2017, mezi lety 1936 a 1967 je počet publikací na rok menší než 1 500 a v grafu tedy nejdou ani vidět. V dnešní době obsahuje 3 673 642 článků, 1 863 745 autorů, 5 022 konferencí a 1 513 časopisů. Všechna data jsou dostupná ve formátu XML.



Obrázek 8: Počet publikací na rok

6.1.1 Analýza DBLP

Pro účely této práce jsme se zaměřili pouze na údaje o autorech kteří spolupracovali na nějakém článku. Na základě toho jsme vytvořili sítě, kde autoři představovali uzly a jejich spolupráce hrany mezi těmito uzly. Při analýze jsme se zaměřili pouze na každý druhý rok v období od roku 1997 až do roku 2015 a vytvořili 10 sítí. Dřívější období neobsahuje tolik záznamů a v

[1] https://gephi.org/

letech 2016 a 2017 ještě mnoho publikaci chybí. Jak je vidět z grafu 7 je pro nás nejzajímavější období přibližně za posledních 20 let. Od roku 2000 počet publikací rapidně stoupá.

Nejprve bylo nutné data ve formátu XML rozparsovat. K tomu využijeme vlastní nástroj, který vygeneruje soubor informací o roku vydání díla, názvu publikace a jména autorů kteří na ní pracovali ve formátu "rok#název díla#autor1#autor2#..."oddělené znakem #. Tento soubor dále zpracujeme pomocí dalšího vlastního nástroje a vygenerujeme síť spoluautorů. Protože dat je mnoho, textový soubor má skoro 200MB, rozdělíme data do 10 souboru, vždy pro dané časové období. Jednotlivé řádky filtrujeme pomocí nástroje grep, příkaz potom vypadá následovně:

```
cat dblp.txt | grep '^1997' > dblp-1997.txt
```

V tabulkách 5 a 6 jsou výsledky analýz globálních vlastností. Všechny sítě nejsou příliš husté, u sítě z roku 1997 je průměrný stupeň pouze 2.846 a hustota sítě je také velmi malá. S každým dalším rokem se ale průměrný stupeň zvětšoval. Paradoxně ale hustota klesala. Je to dáno tím, že s každým novým uzlem se maximální počet hran zvýší o *n - 1*. Zajímavé také je, jak se zvětšila největší komponenta souvislosti. V prvním případě obsahuje asi jen 10% všech uzlů a v posledních dvou případech to už je víc než polovina všech uzlů. Průměrný shlukovací koeficient byl ve všech případech víc než 0,5 a postupem času ještě rostl. Tedy pravděpodobnost, že dva sousedé daného uzlu budou také spojeni hranou, byla poměrně vysoká. Modularita se ve všech případech blížila jedničce, což značí, že uzly mají tendenci se spojovat s uzly v rámci své komunity. Taktéž asortativita je poměrně vysoká, to znamená, že uzly mají tendenci se spojovat s uzly, které jsou v nějakém ohledu podobné.

	Počet uzlů	Počet hran	Průměrný stupeň	Komponent souvislosti	Největší komponenta	Hustota sítě	Mocninný exponent
1997	30816	43854	2.846	7524	3012	9.236358e-5	2.404
1999	39014	60074	3.079	9280	3654	7.893810e-5	2.291
2001	49906	78963	3.165	10987	6913	6.340986e-5	2.78
2003	64900	121377	3.740	12723	13700	5.763465e-5	1.844
2005	89864	175825	3.913	15871	26216	4.354557e-5	2.053
2007	115929	230021	3.968	18385	42624	3.42308e-5	2.424
2009	144322	307209	4.257	20684	62026	2.949863e-5	2.269
2011	174030	402048	4.620	22492	85532	2.654986e-5	2.177
2013	201120	527444	5.245	23433	109185	2.607942e-5	2.244
2015	224119	642744	5.736	23899	130580	2.559249e-5	1.948

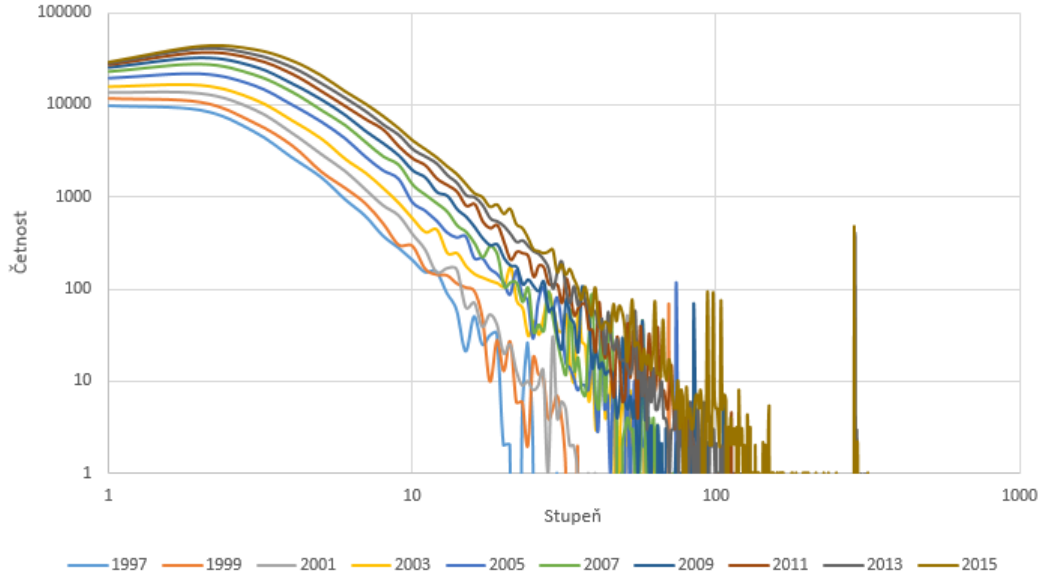
Tabulka 5: Výsledky analýzy, DBLP

	Průměrná délka nejkratší cesty	Průměr sítě	Prům. shluk. koeficient	Modularita	Asortativita
1997	13.318	40	0.591	0.980	0.728
1999	14.390	43	0.602	0.978	0.966
2001	16.940	53	0.629	0.975	0.859
2003	15.267	40	0.655	0.965	0.774
2005	13.577	41	0.679	0.960	0.844
2007	12.024	42	0.690	0.946	0.730
2009	10.878	33	0.702	0.931	0.744
2011	9.703	34	0.72	0.908	0.611
2013	8.743	37	0.731	0.887	0.980
2015	7.987	33	0.74	0.862	0.941

Tabulka 6: Výsledky analýzy, DBLP

Distribuce stupňů

Obrázek 9 zobrazuje graf distribuce stupňů. Osy jsou v logaritmickém měřítku, neboť nejvíce uzlů je s malým stupněm a naopak s velkým stupněm je jen několik málo uzlů. Několik ojedinělých uzlů je vysokého stupně. To odpovídá publikacím, na kterých společně pracovalo mnoho autorů. Funkce odpovídá mocninnému rozdělení. Do roku 2001 je nejvíce uzlů s jedním stupněm a z tabulky 5 je vidět, že i průměrný stupeň je poměrně malý. Jak ale rostl počet autorů a počet publikací, rostl i průměrný stupeň a od roku 2003 je už nejvíc uzlů se stupněm 2.

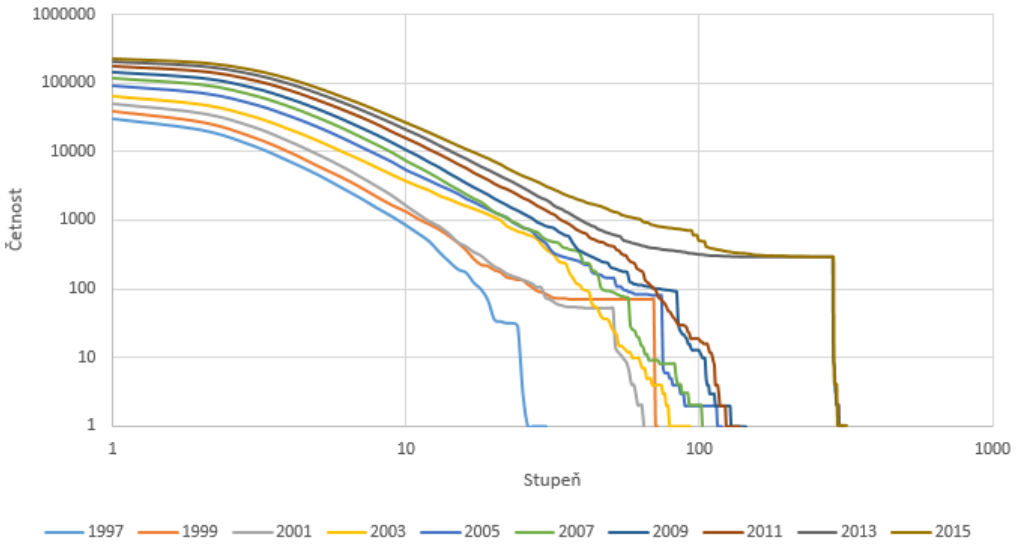


Obrázek 9: Distribuce stupňů, DBLP

Kumulativní distribuce stupňů

Kumulativní distribuční funkci jsem použil pro výpočet mocninného exponentu, který je v tabulce výše a je potřebný pro výpočet mocninného rozdělení. Toto mocninné rozdělení ale neplatí

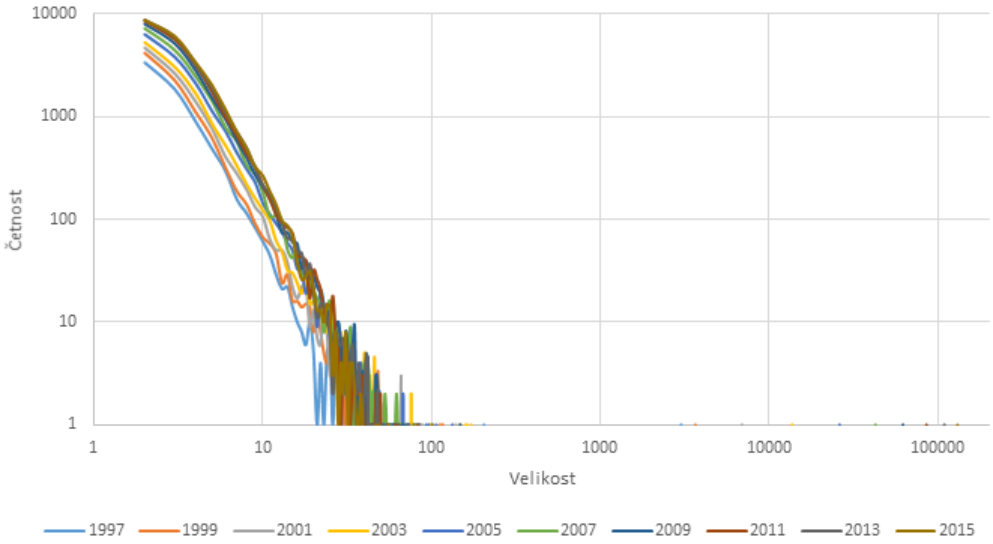
pro všechny stupně, ale přibližně od stupně 5 po stupeň 50. Grafy těchto funkcí jsou na obrázku 10.



Obrázek 10: Kumulativní distribuce stupňů, DBLP

Distribuce velikosti komponent souvislosti

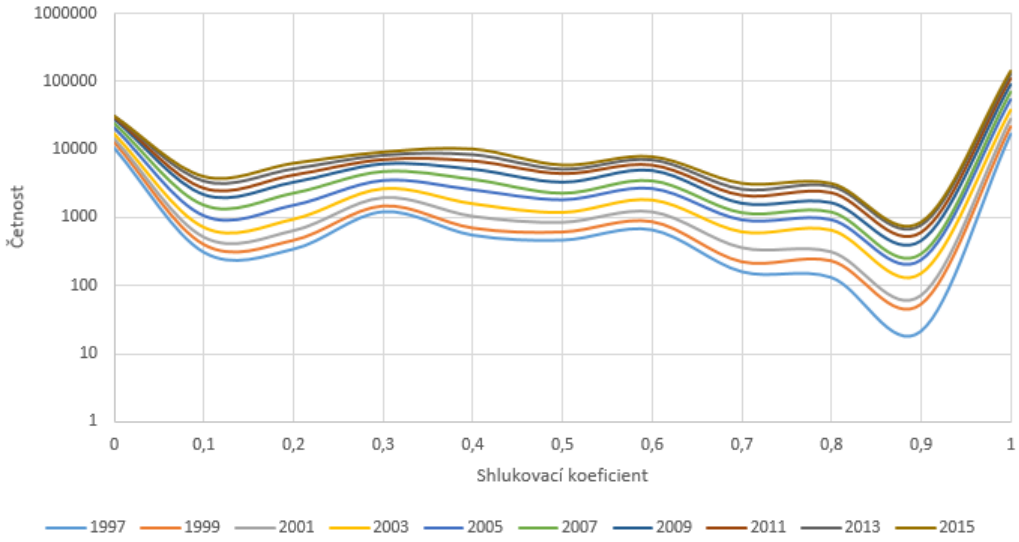
Na obrázku 11 je graf distribuce velikosti komponent souvislosti. Graf zachycuje pouze velikosti od 2, protože žádná komponenta neobsahovala pouze jeden uzel, tedy žádný uzel nebyl osamocený. Největší komponenty jednotlivých sítí byly jednoznačně větší než všechny ostatní, u sítí z let 2013 a 2015 obsahovaly víc než polovinu všech uzlů, a osamostatnily se v pravé části grafu.



Obrázek 11: Distribuce velikosti komponent souvislosti, DBLP

Distribuce shlukovacího koeficientu

Protože shlukovací koeficient je reálné číslo od nuly do jedné, rozdělil jsem tento interval na 10 intervalů a pro každý jsem spočítal počet hodnot shlukovacího koeficientu, které se do tohoto intervalu vešly. Výsledek můžeme vidět na obrázky 12.



Obrázek 12: Distribuce shlukovacího koeficientu, DBLP

6.1.2 Analýza největších komponent dblp

Sítě DBLP jsou nesouvislé. Rozhodl jsem se tedy analyzovat jejich největší komponenty souvislosti. Kromě centralit jsem analyzoval i ostatní vlastnosti. Kromě počtu uzlů a počtu hran jsou hodnoty velmi podobné hodnotám celých sítí. Pochopitelně se zvýšil průměrný stupeň, ale ostatní vlastnosti se zvýšily jen nepatrně. I zde jsou sítě velmi řídké. Výsledky jsou v tabulkách 7 a 8.

	Počet uzlů	Počet hran	Průměrný stupeň	Prům. délka nejkratší cesty	Průměr sítě	Prům. shluk. koeficient
1997	3012	6126	4.068	13.556	40	0.609
1999	3654	8144	4.458	14.642	43	0.642
2001	6913	16089	4.655	17.048	53	0.658
2003	13700	41327	6.033	15.299	40	0.712
2005	26216	74732	5.701	13.585	41	0.72
2007	42624	117196	5.499	12.027	42	0.724
2009	62026	176797	5.701	10.88	33	0.732
2011	85532	262983	6.149	9.703	34	0.748
2013	109185	381774	6.993	8.743	37	0.753
2015	130580	491467	7.528	7.987	33	0.758

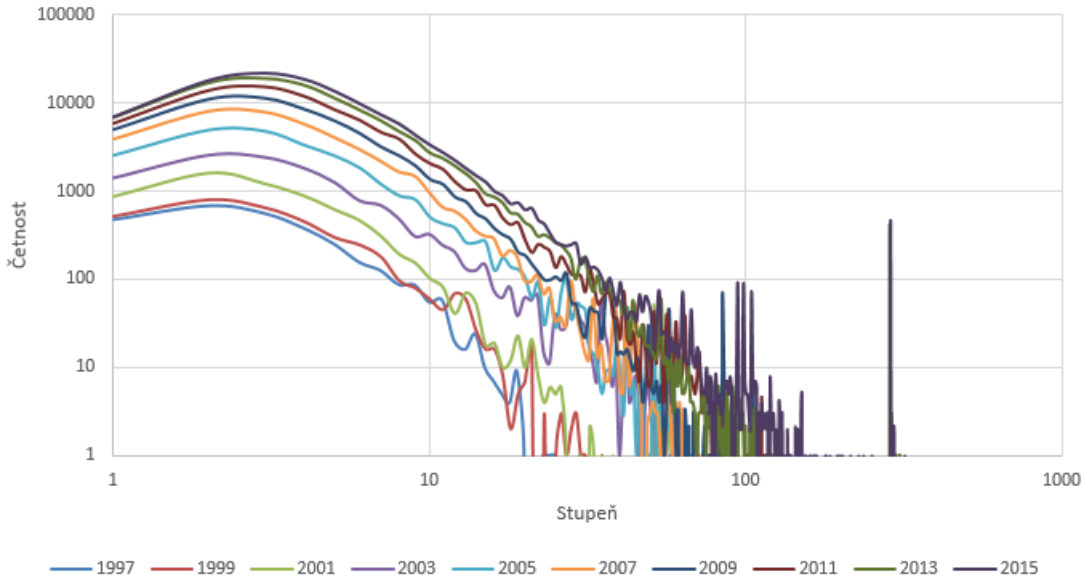
Tabulka 7: Výsledky analýzy největších komponent, DBLP

	Hustota sítě	Mocninný exp.	Modularita	Asortativita
1997	1.350956e-3	-3	0.875	0.214
1999	1.220252e-3	-2.724	0.885	0.447
2001	6.734242e-4	-2.258	0.886	0.864
2003	4.404072e-4	-1.612	0.9	0.701
2005	2.174804e-4	-1.93	0.904	0.671
2007	1.290162e-4	-2.196	0.946	0.730
2009	9.191033e-5	-2.126	0.876	0.712
2011	7.189613e-5	-2.134	0.858	0.563
2013	6.40493e-5	-2.088	0.838	0.979
2015	5.764669e-5	-1.856	0.862	0.941

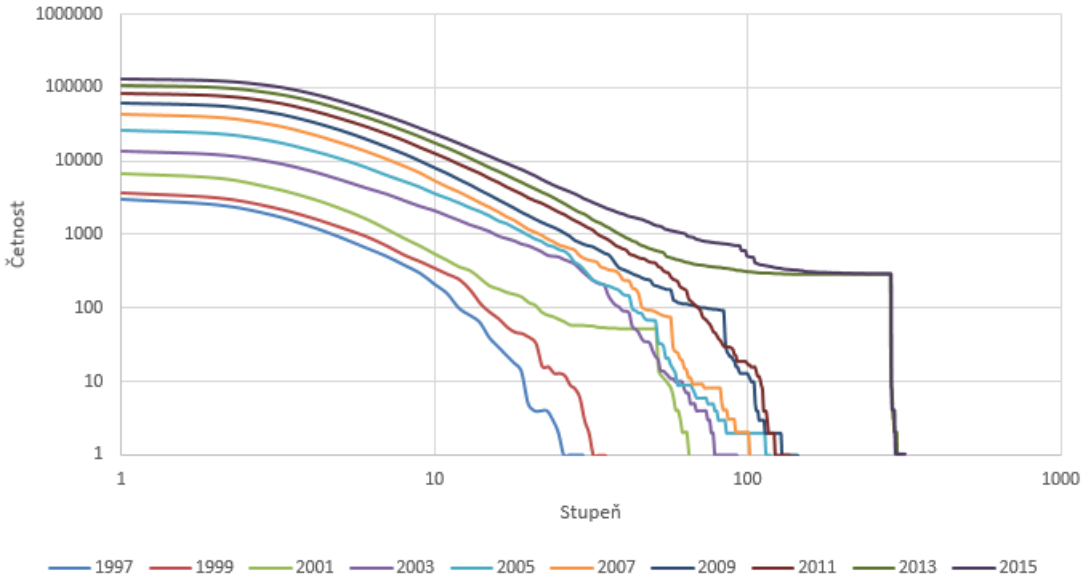
Tabulka 8: Výsledky analýzy největších komponent, DBLP

Grafy distribucí

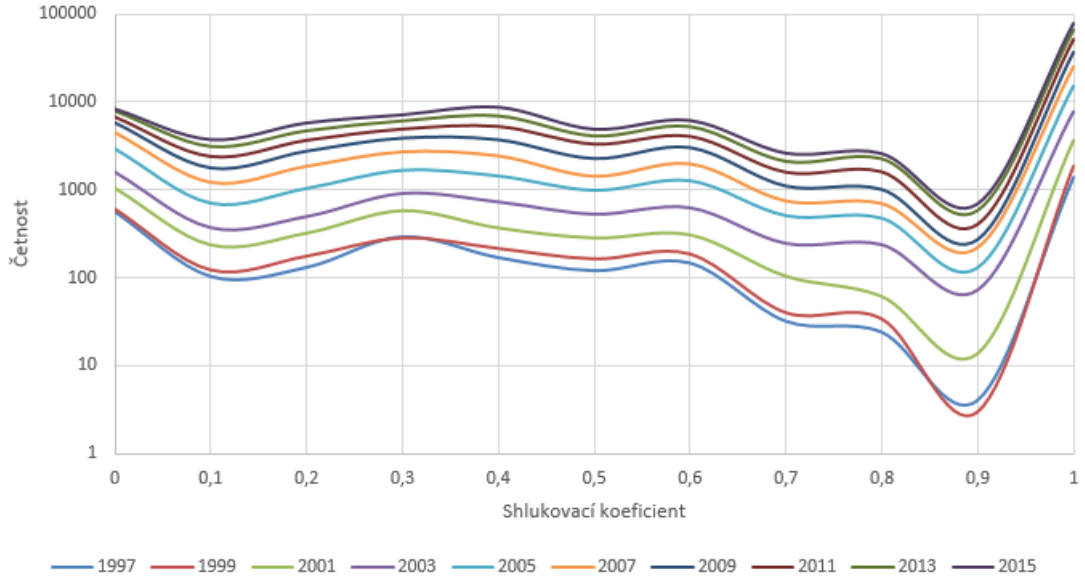
Na obrázcích 13, 14 a 15 můžeme vidět grafy distribucí. Stejně jako vypočítané hodnoty jsou i grafy jednotlivých distribucí velmi podobné původním sítím. Stejně jako v předchozím případě jsem použil kumulativní distribuci stupňů pro výpočet mocninného exponentu.



Obrázek 13: Distribuce stupňů, DBLP



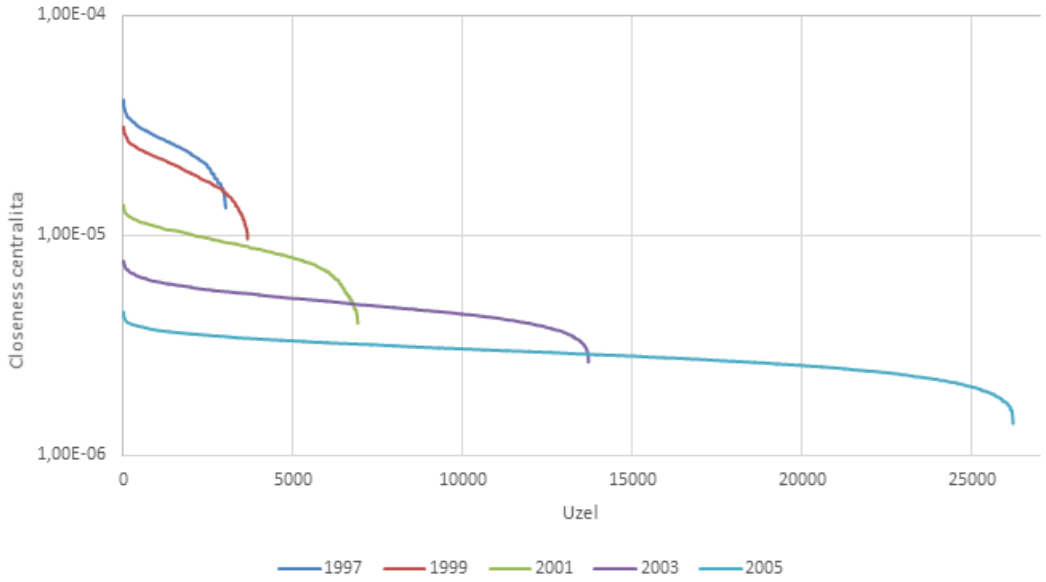
Obrázek 14: Kumulativní distribuce stupňů, DBLP



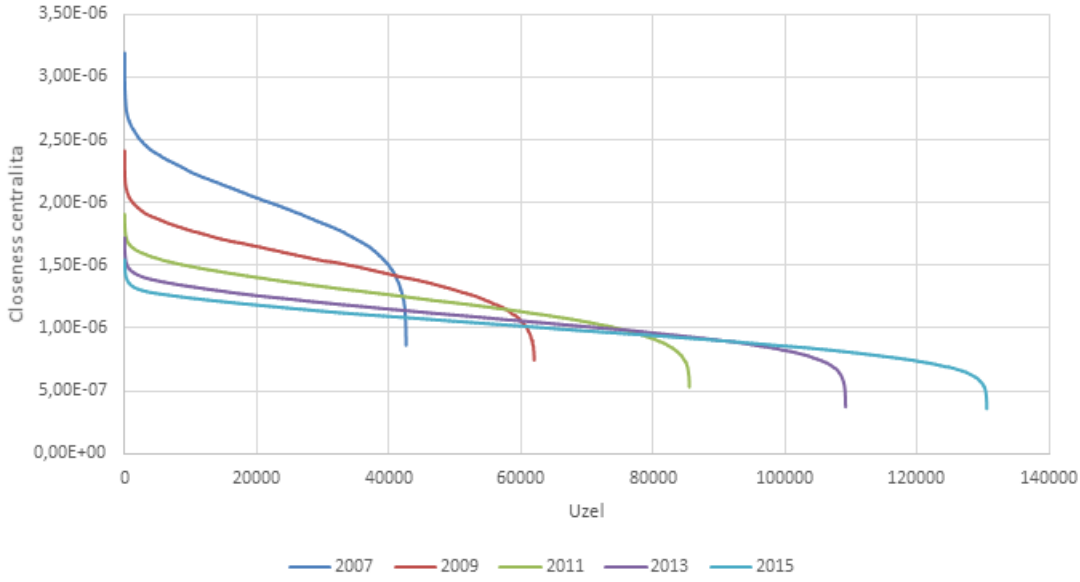
Obrázek 15: Distribuce shlukovacího koeficientu, DBLP

Closennes centralita

Grafy centralit jsem rozdělil na 2, první reprezentuje období od roku 1997 do roku 2005 a druhý období od roku 2007 do roku 2015. Důvodem byl velký rozdíl v počtech uzlů v uvedených obdobích. Pokud bychom vykreslili grafy centralit pro celé období od roku 1997 do roku 2015, byly by nepřehledné. Closeness centralita pro období od roku 1997 do roku 2005 je na obrázku 16, closeness centralita pro období od roku 2007 do roku 2015 je na obrázk 17. Všechny centrality jsou si velice podobné. V každé síti existuje několik uzlů s výrazně vyšší centralitou a několik uzlů s výrazně menší centralitou.



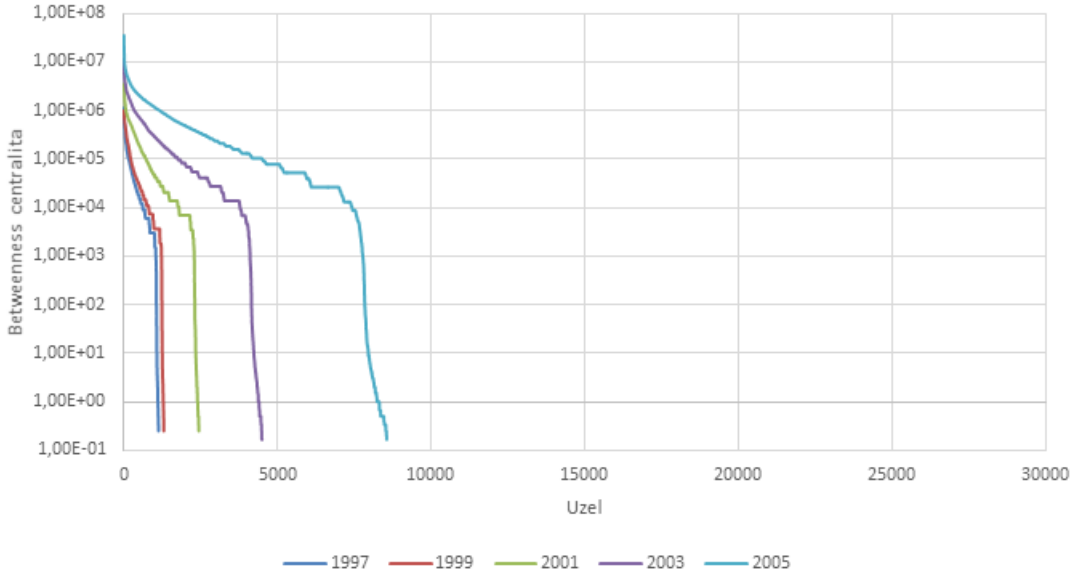
Obrázek 16: Closeness centralita, DBLP



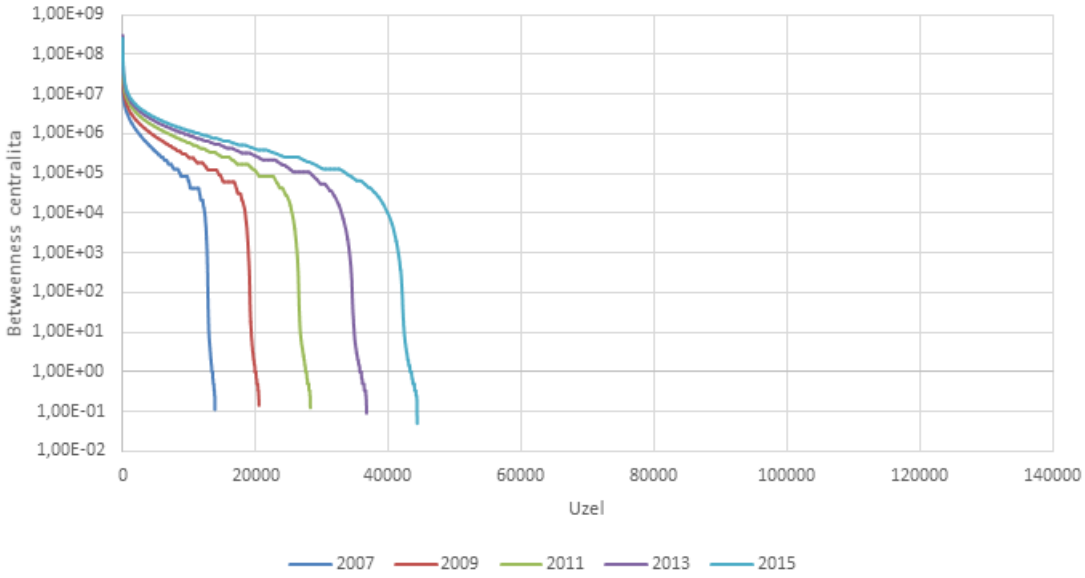
Obrázek 17: Closeness centralita, DBLP

Betweenness centralita

I graf betweenness centrality byl rozdělen na 2, pro dvě období, aby byly centrality lépe patrné. I v tomto případě jsou si všechny centrality navzájem velmi podobné. Několik uzlů má výrazně vyšší centralitu, ale u všech sítí od určitého uzlu centralita prudce klesá a většina uzlů má centralitu nulovou.



Obrázek 18: Betweenness centralita, DBLP



Obrázek 19: Betweenness centralita, DBLP

6.2 High School

Tato datová sada obsahuje temporální sít kontaktů mezi studenty devíti tříd střední školy ve francouzském Marseille po dobu 5 dní v prosinci 2013. Dataset obsahuje seznam hran představujících kontakt dvou lidí s časovým údajem po dvaceti sekundách ve kterém byli v kontaktu. Data jsou dostupná z webových stránek SocioPatterns [10].

6.2.1 Analýza High School

Síť jsem analyzoval jako temporální a poté jsem ji rozdělil do 10 časových oken, která jsem analyzoval jako jednoduché sítě. Ve dvou časových oknech nebyly žádné hrany aktivní a tedy výsledné sítě neobsahovaly žádné uzly ani hrany a nebylo tedy co analyzovat. Bohužel se mi nepovedlo spočítat temporální betweenness centralitu. Výpočet nejkratších cest trval příliš dlouho.

V následující tabulce 9 jsou výsledky analýzy pro celou temporální síť. Průměrná latence nejkratší cesty a průměru sítě respektující čas je v hodinách. Průměrný stupeň je poměrně vysoký, ale je to dáno tím, že se jedná o celou temporální síť. Jakmile se síť rozdělila na časová okna, došlo ke snížení hustoty sítě a poměrně výrazně se snížil i průměrný stupeň.

Počet uzlů	327	Prům. délka nejkratší cesty	2.159
Počet hran	5818	Průměr sítě	4
Průměrný stupeň	35.584	Prům. délka nejkratší temp. cesty	4.969
Komponent souvislosti	1	Průměr sítě respektující čas	94.156
Hustota sítě	0.109	Modularita	0.576
Prům. shluk. koeficient	0.101	Asortativita	0.033

Tabulka 9: Výsledky analýzy, High School

V tabulkách 10 a 11 jsou výsledky globálních vlastností jednotlivých časových oken. Čtvrté a deváté časové okno neobsahuje žádné aktivní hrany a i druhé časové okno obsahuje velmi málo uzlů. Odhadoval bych, že půjde o večerní, nebo noční hodiny, kdy studenti v kontaktu nejsou.

Časové okno	Počet uzlů	Počet hran	Průměrný stupeň	Komponent souvislosti	Největší komponenta	Hustota sítě
1	312	2242	14.372	1	312	0.046
2	188	215	2.287	31	52	0.012
3	310	2548	16.439	1	310	0.053
5	300	1964	13.093	1	300	0.044
6	205	454	4.429	9	171	0.022
7	261	1086	8.321	2	259	0.032
8	287	1610	11.220	2	285	0.039
10	299	2075	13.880	1	299	0.046

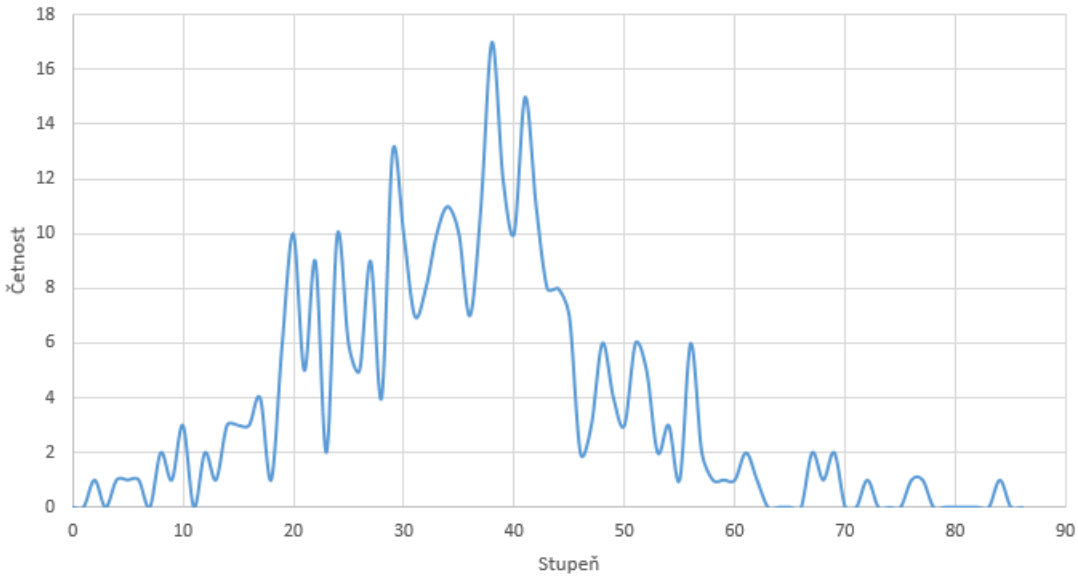
Tabulka 10: Výsledky analýzy časových oken, High School

Časové okno	Průměrná délka nejkratší cesty	Průměr garfu	Prům. shluk. koeficient	Modularita	Asortativita
1	2.901	5	0.401	0.683	0.092
2	4.125	11	0.186	0.851	0.402
3	2.775	6	0.417	0.671	0.115
5	3.216	7	0.443	0.719	0.094
6	4.071	9	0.324	0.71	0.148
7	3.548	9	0.325	0.671	0.067
8	3.097	7	0.384	0.677	0.116
10	3.075	6	0.434	0.714	0.193

Tabulka 11: Výsledky analýzy časových oken, High School

Distribuce stupňů

Na obrázku 20 je graf distribuce stupňů pro celou temporální síť. Tento graf neodpovídá mocninnému rozdělení, ale spíše připomíná normální rozdělení a proto jsem ani mocninný exponent nepočítal. Na obrázku 21 jsou grafy distribucí stupňů pro jednotlivá časová okna.



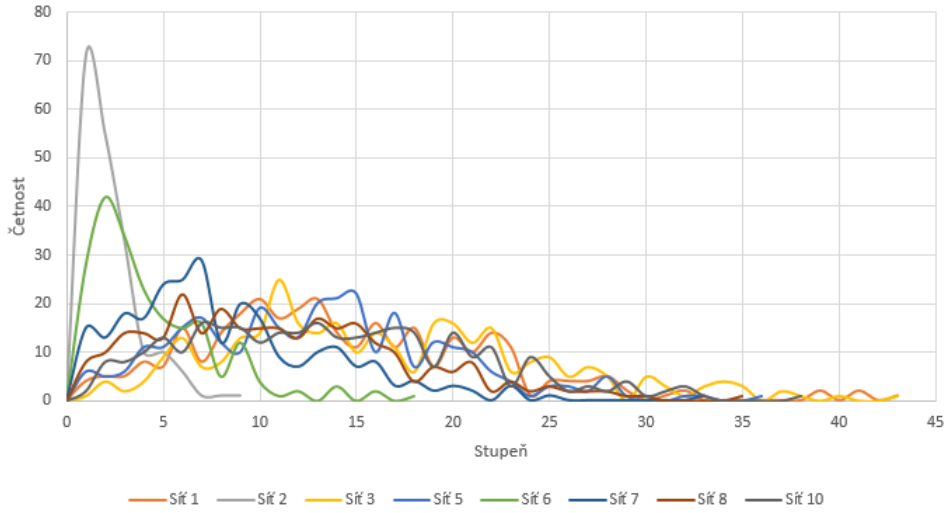
Obrázek 20: Distribuce stupňů, High School

Kumulaticní distribuce stupňů

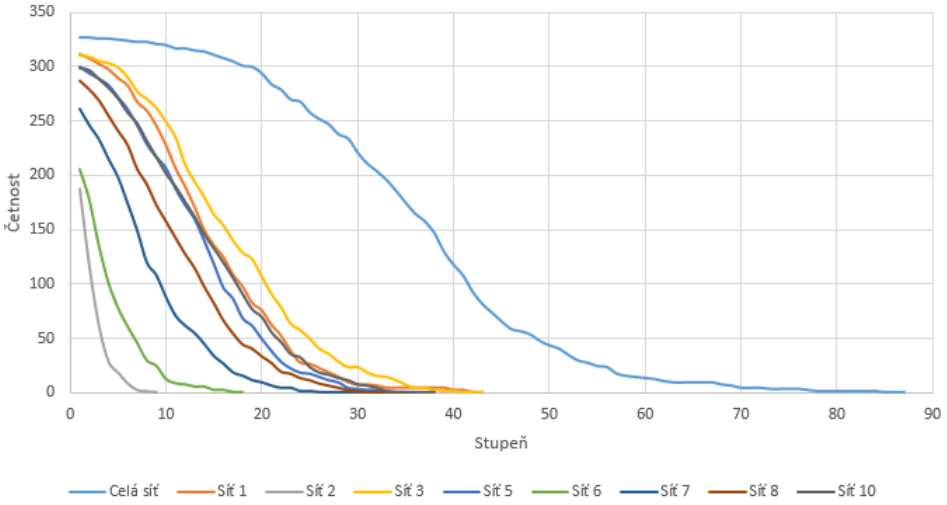
Obrázek 22 představuje graf kumulativní distribuce stupňů.

Distribuce shlukovacího koeficientu

Na obrázku 23 můžeme vidět distribuci shlukovacího koeficientu. Stejně jako u analýzy DBLP jsem interval od nuly do jedné rozdělil na 10 intervalů a pro každý jsem spočítal počet hodnot, které se do něj vešly.



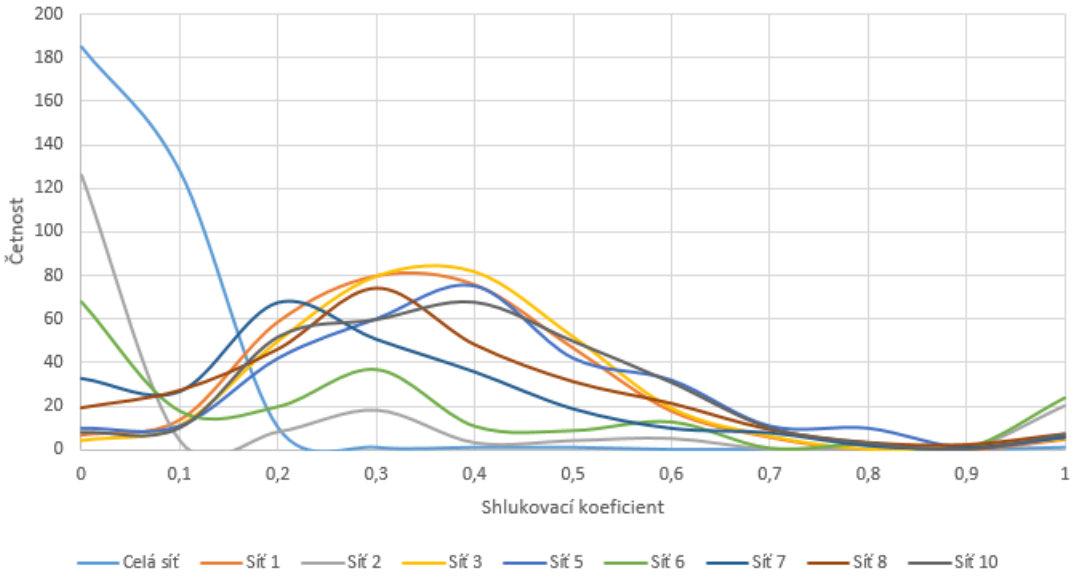
Obrázek 21: Distribuce stupňů pro časová okna, High School



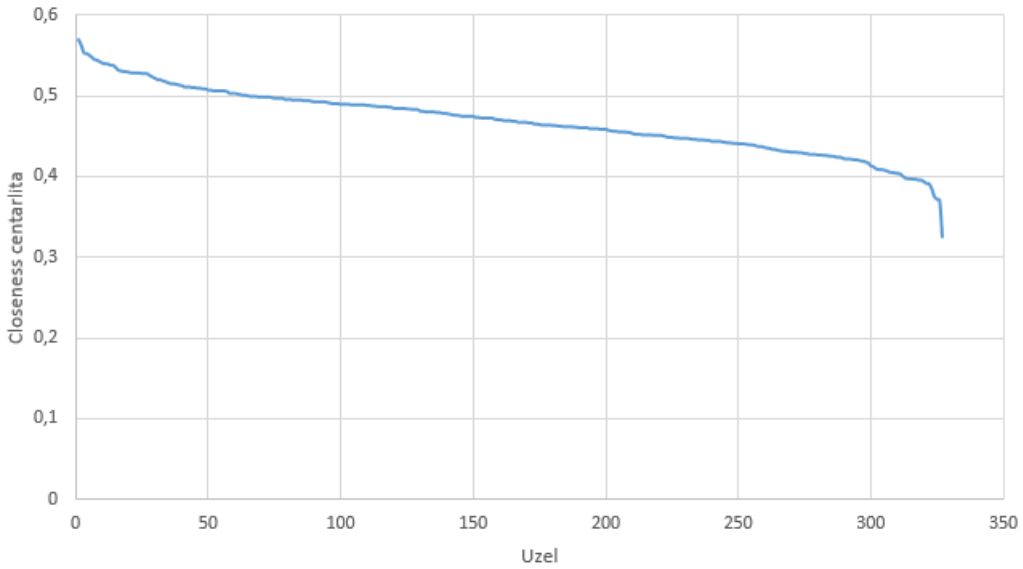
Obrázek 22: Kumulativní distribuce stupňů, High School

Closeness centralita

Pro tuto datovou sadu jsem spočítal několik closeness centralit. Na obrázku 24 je closeness centralita celé sítě, na obrázku 25 je spočtena temporální closeness centralita celé sítě a na obrázku 26 jsou closeness centrality jednotlivých časových oken. Dvě časová okna zde chybí, jsou to ty ve kterých nebyly žádné hrany aktivní. U oken, ve kterých bylo více komponent souvislosti, jsem spočítal centarlity pouze pro tu největší z nich. Grafy centralit jsou podobné síti DBLP. I zde je několik uzlů s výrazně vyšší, nebo nižší centralitou. Vyjímkou je temporální closeness centralita, která má téměř lineární průběh a ke konci pravé strany poměrně strmě klesá k nulovým hodnotám.



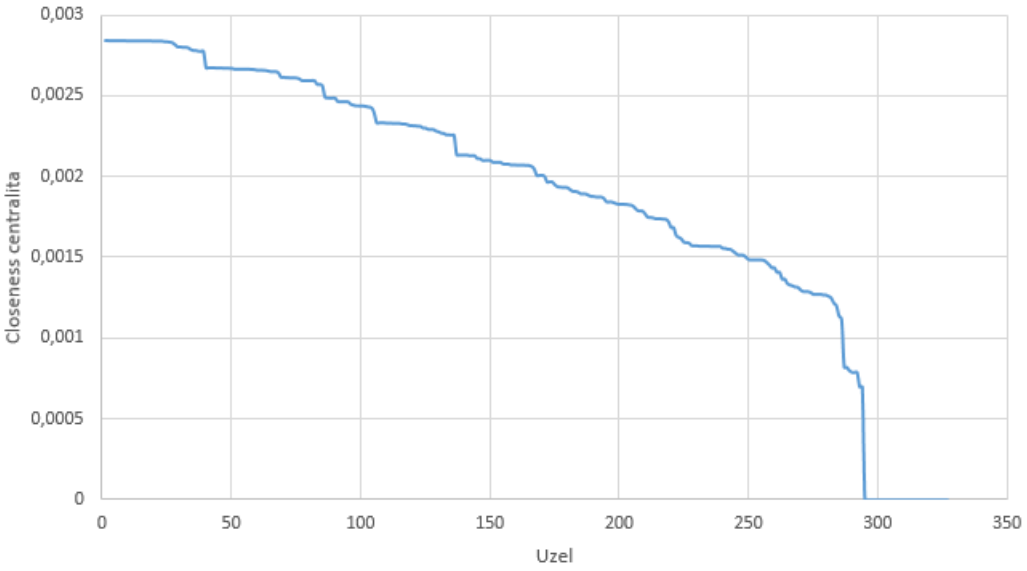
Obrázek 23: Distribuce shluovacího koeficientu, High School



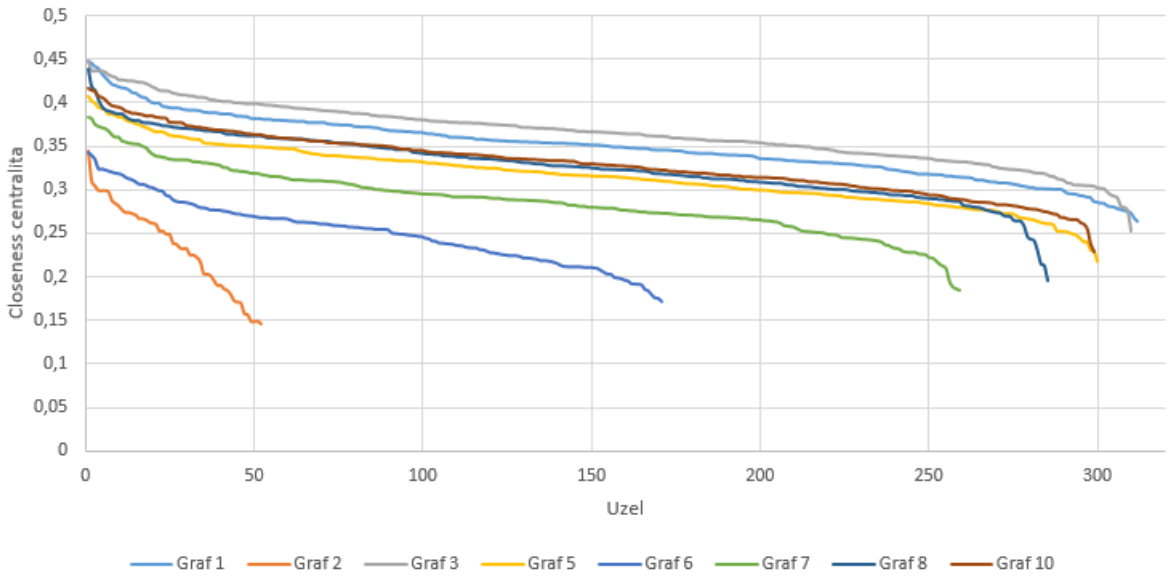
Obrázek 24: Closeness centralita, High School

Betweenness centralita

Na obrázcích 27 a 28 jsou grafy betweenness centralit pro celou síť a pro jednotlivá časová okna. Bohužel zde chybí temporální betweenness centralita, protože její výpočet byl příliš časově náročný. Křivky jednotlivých betweenness centralit jsou si v tomhle případě velmi podobné s closeness centralitami.



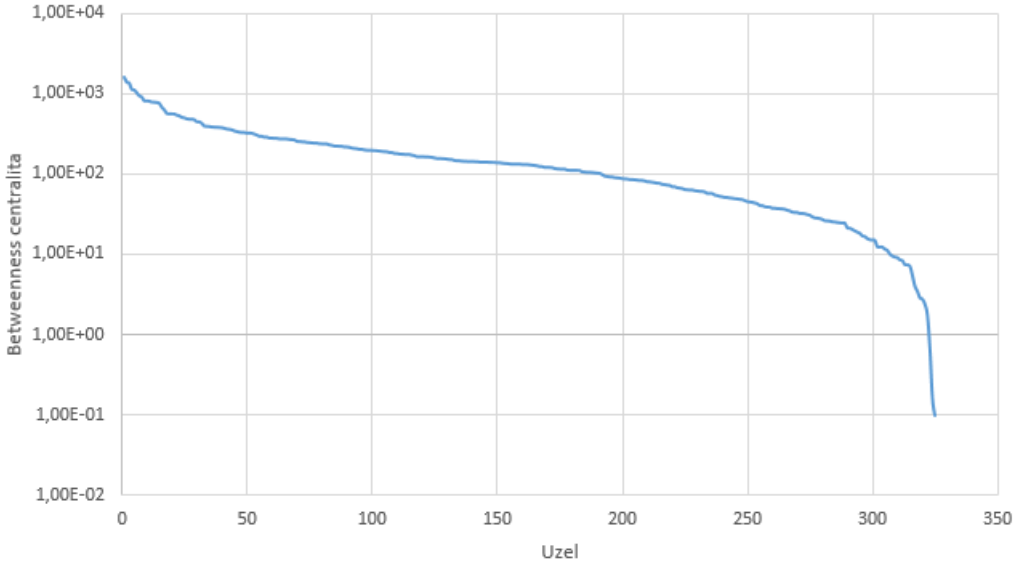
Obrázek 25: Temporální closeness centralita, High School



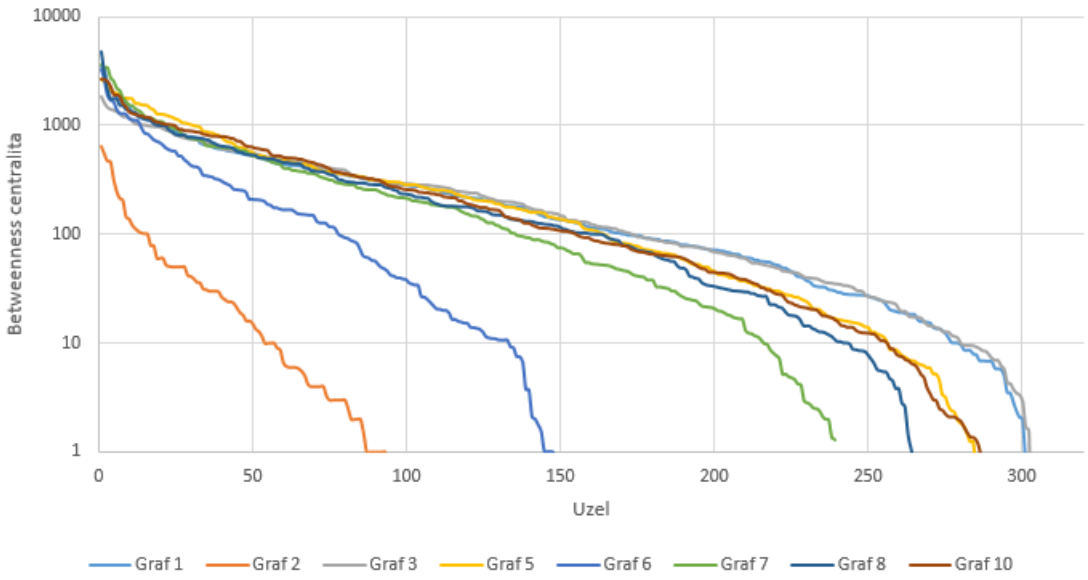
Obrázek 26: Closeness centralita pro časová okna, High School

6.3 US Flights

Síť USflights.net [1] obsahuje temporální síť letů mezi letišti v USA. Obsahuje 2126 uzlů představujících jednotlivá letiště, 5629896 letů a 493 leteckých společností představujících hrany. Data jsou sbírána po měsících od ledna 1990 až do června 2014. Hrany jsou ohodnoceny počtem pasažérů přepravených za jeden měsíc.



Obrázek 27: Betweenness centralita, High School



Obrázek 28: Betweenness centralita pro časová okna, High School

6.3.1 Analýza US Flights

Pro analýzu jsem datovou sadu upravil a informace o leteckých společnostech a pasažérech jsem odstranil. Měl jsem tak pouze temporální síť letišť a letů mezi nimi po jednotlivých měsících, čímž se celá síť zmenšila na počet hran 49516.

V tabulce 12 jsou výsledky analýzy pro celou temporální síť. Průměr a průměrná latence nejkratší cesty respektující čas jsou v měsících.

V tabulkách 13 a 14 jsou výsledky analýzy pro jednotlivá časová okna, ze kterých je vidět jak letecká doprava houstla. Na rozdíl od předchozí sítě, nedošlo po rozdělení k tak velkému

Počet uzlů	2126	Prům. délka nejkratší cesty	2.816
Počet hran	49516	Průměr sítě	7
Průměrný stupeň	35.584	Prům. délka nejkratší temp. cesty	162.164
Komponent souvislosti	3	Průměr sítě respektující čas	294
Největší komponenta	2122	Hustota sítě	0.109
Prům. shluk. koeficient	0.504	Asortativita	0.0045
Mocninný exponent	0.95	Modularita	0.224

Tabulka 12: Výsledky analýzy, US Flights

poklesu průměrného stupně a ve všech případech se pohybuje kolem 30.

Časové okno	Počet uzlů	Počet hran	Průměrný stupeň	Komponent souvislosti	Největší komponenta	Hustota sítě	Mocninný exponent
1	459	6927	30.183	1	459	0.066	1.22
2	474	6915	29.177	1	474	0.062	1.12
3	573	8094	28.251	1	573	0.049	1.06
4	559	8610	30.805	2	557	0.055	1
5	942	12756	27.082	1	942	0.029	0.94
6	1435	22843	31.836	2	1433	0.022	0.98
7	1492	24617	32.998	3	1488	0.022	0.92
8	1474	24723	33.546	3	1470	0.023	0.84
9	1515	23552	31.092	3	1511	0.021	0.9
10	1407	21047	29.918	1	1407	0.021	0.86

Tabulka 13: Výsledky analýzy časových oken, US Flights

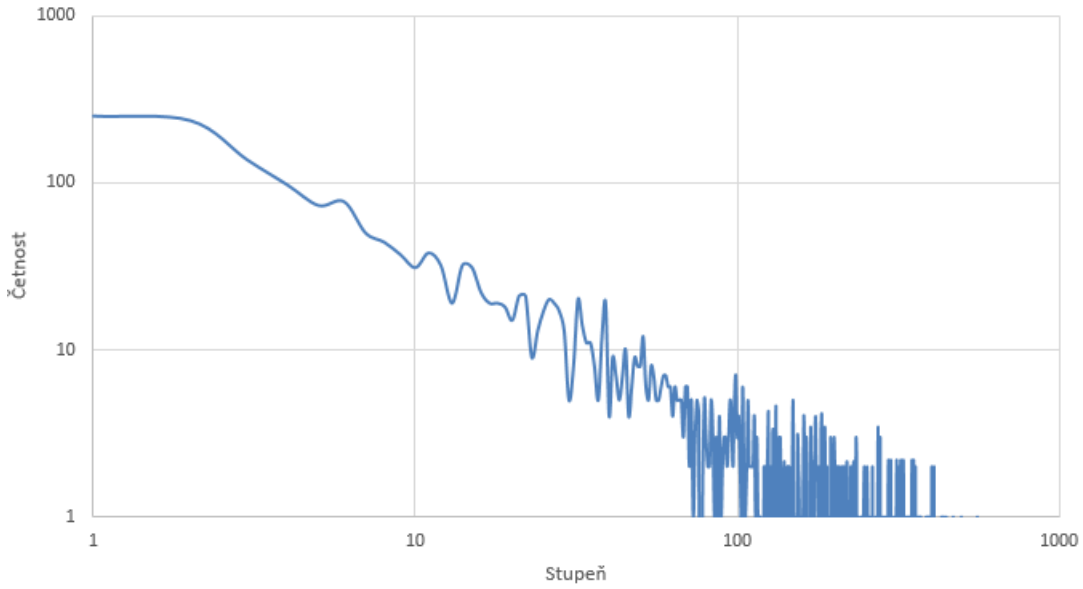
Časové okno	Průměrná délka nejkratší cesty	Průměr garfu	Prům. shluk. koeficient	Modularita	Asortativita
1	2.582	5	0.606	0.063	-0.027
2	2.511	6	0.562	0.112	-0.042
3	2.628	6	0.579	0.116	0.0034
4	2.556	6	0.582	0.05	-0.052
5	3.062	8	0.584	0.273	0.086
6	2.975	7	0.584	0.294	0.046
7	2.943	7	0.553	0.273	0.058
8	2.942	6	0.541	0.257	0.076
9	2.955	7	0.547	0.273	0.054
10	2.966	7	0.550	0.277	0.066

Tabulka 14: Výsledky analýzy časových oken, US Flights

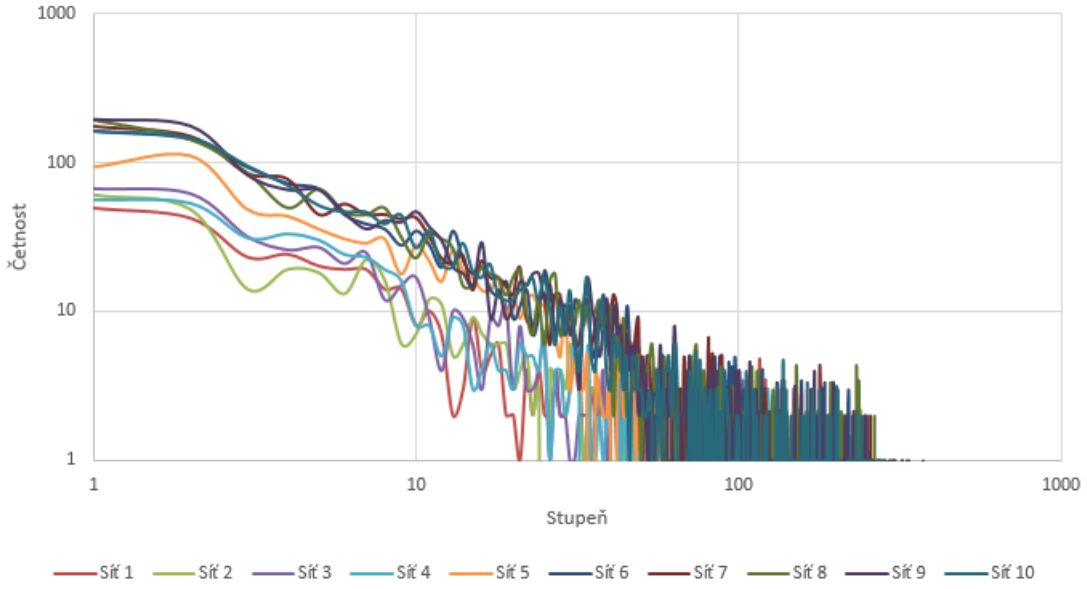
Grafy distribucí

Na následujících obrázcích 29, 30, 31 a 32 můžeme vidět distribuce pro sít US Flights. Z grafů distribucí ze zdálo, že by mohly odpovídat mocninnému rozdělení. Mocninný exponent, vypo-

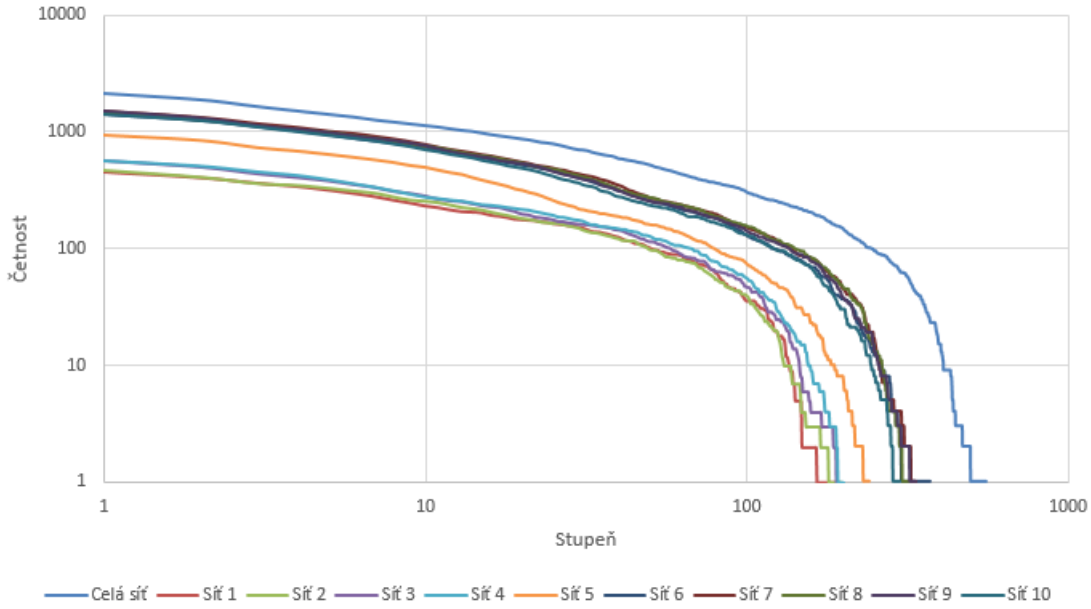
čítaný z kumulativní distribuce v intervalu přibližně 30 až 100 stupňů, ale vycházel přibližně 1. V tomto intervalu tedy grafy distribucí mocninnému rozdělení moc neodpovídají.



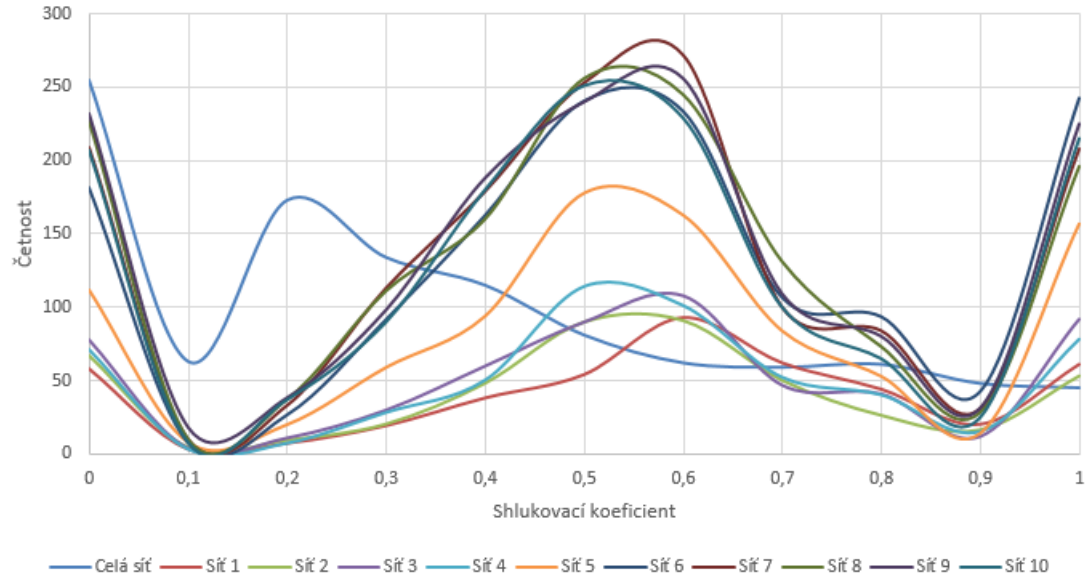
Obrázek 29: Distribuce stupňů pro celou síť, US Flights



Obrázek 30: Distribuce stupňů pro časová okna, US Flights



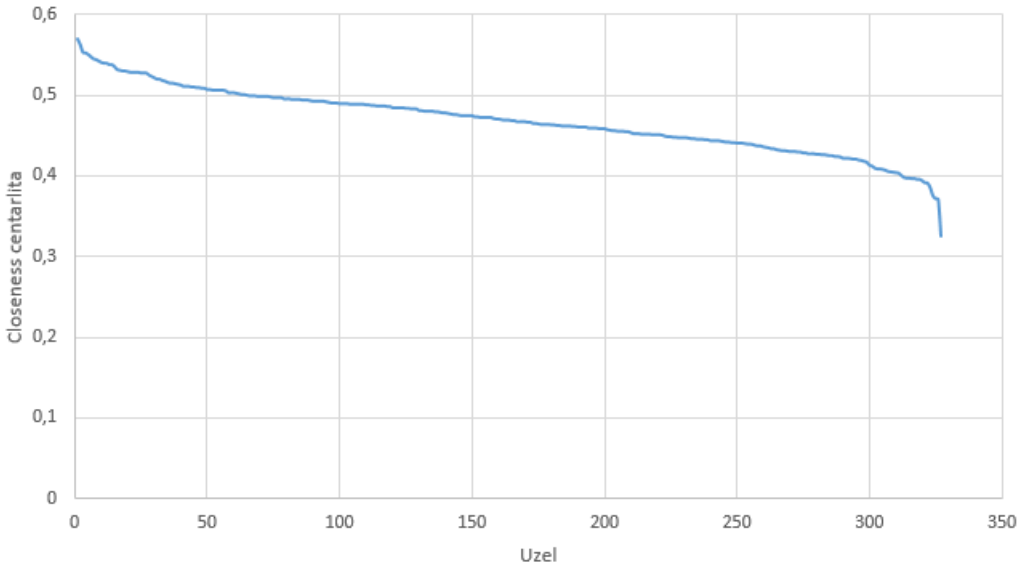
Obrázek 31: Kumulativní distribuce stupňů, US Flights



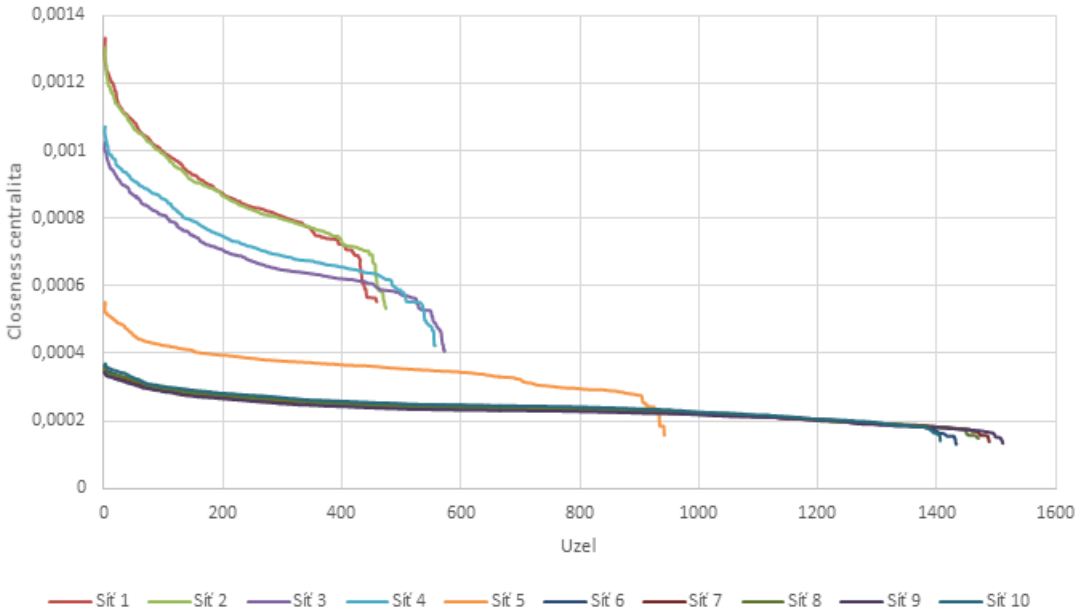
Obrázek 32: Distribuce shluovacího koeficientu, US Flights

Closeness centralita

Stejně jako v předchozím případě, jsem i zde počítal několik closeness centralit. Na Obrázku 33 je closeness centralita pro celou temporální síť. Na obrázku 34 jsou jednotlivá časová okna a obrázek 35 zobrazuje temporální closeness centralitu. Podobně jako u sítě High School se i tato temporální closeness centralita trochu liší od centralit jednoduchých sítí.



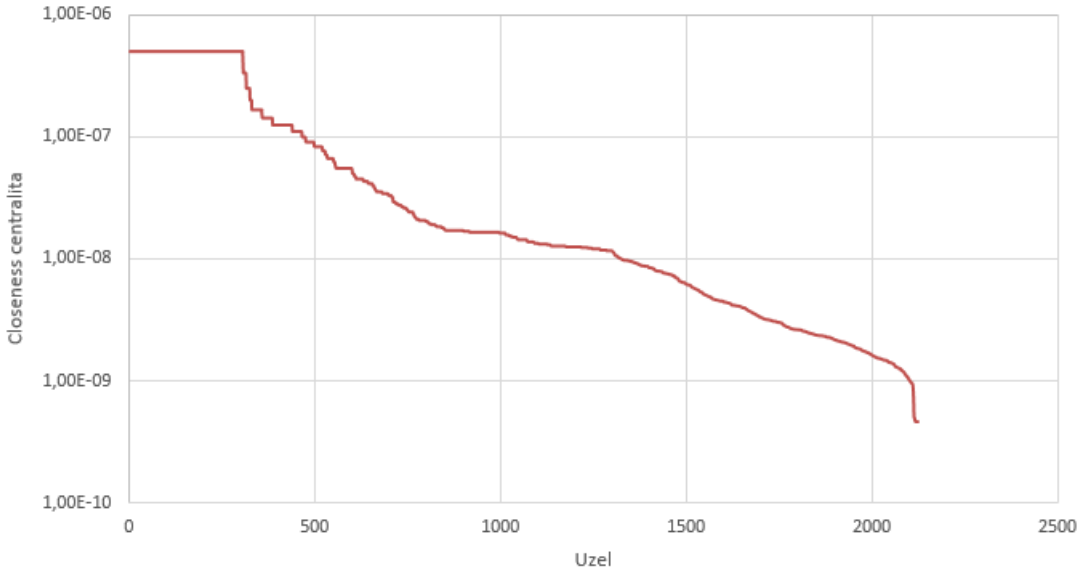
Obrázek 33: Closeness centralita, US Flights



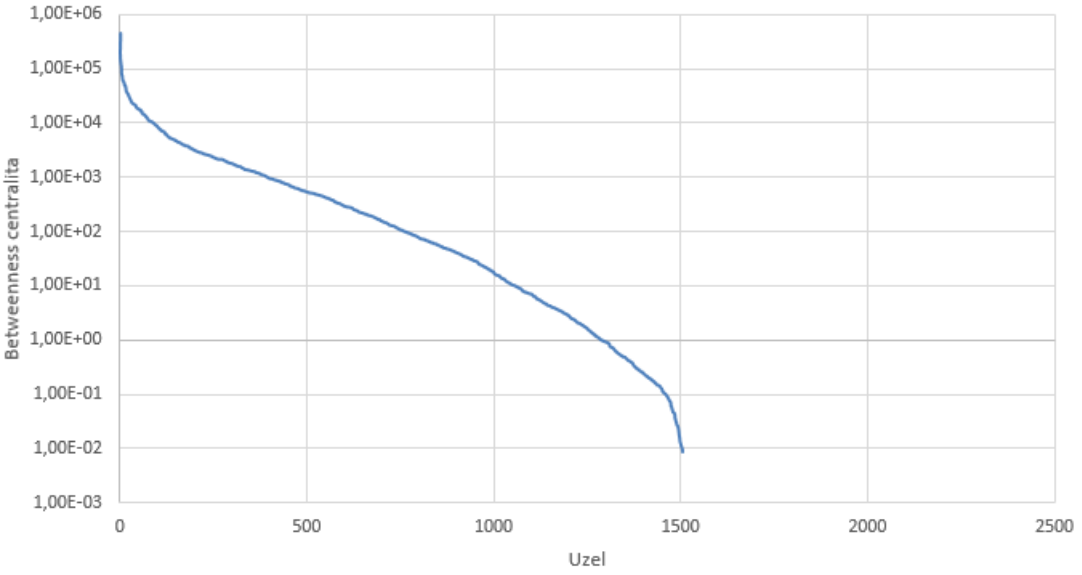
Obrázek 34: Closeness centralita časových oken, US Flights

Betweenness centralita

Podobně jako u sítě High school i zde chybí temporální betweenness centralita, protože ji kvůli časové náročnosti výpočtu nebylo možné spočítat. Betweenness centralita pro celou temporální síť a pro jednotlivé časové okna jsou na obrázcích 36 a 37.



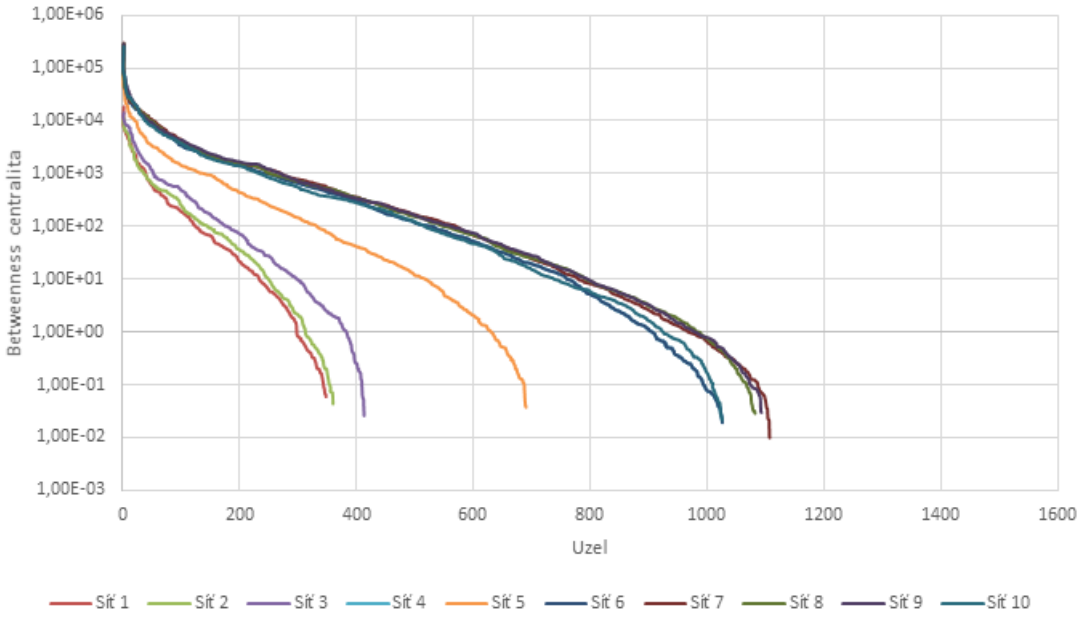
Obrázek 35: Temporální closeness centralita, US Flights



Obrázek 36: Closeness centralita, US Flights

6.4 Workplace

Tento dataset obsahuje temporální síť kontaktů mezi jednotlivci v administrativní budově ve Francii, od 24. června do 3. července 2013. Ze všech sítí analyzovaných v této práci je tahle síť nejmenší, obsahuje pouze 92 uzlů a 755 hran. Tuto malou síť jsem vybral protože jsem doufal, že na ní budu schopný spočítat temporální betweenness centralitu. Bohužel i na této síti trval výpočet příliš dlouho. Síť je sice malá, ale obsahuje poměrně dost časových okamžiků, celkem 49381 po 20 sekundách. Proto jsem síť ještě zmenšil a analyzoval jí pouze pro prvních 1812



Obrázek 37: Closeness centralita časových oken, US Flights

časových okamžiků, tedy asi pro prvních 10 hodin. Tím se mi trochu zmenšil i počet uzlů a hran.

6.4.1 Analýza Workplace

V tabulce 15 jsou výsledky analýzy celé temporální sítě. Průměrná latence nejkratší cesty a průměr sítě respektující čas je v hodinách. V tabulkách 16 a 17 jsou výsledky pro časová okna. Časových oken je pouze 5, protože sít byla tak malá, že při rozdělení na 10 časových oken vznikly jen velmi malé sítě.

Počet uzlů	72	Prům. délka nejkratší cesty	2.958
Počet hran	188	Průměr sítě	7
Průměrný stupeň	5.222	Prům. délka nejkratší temp. cesty	3.604
Komponent souvislosti	1	Průměr sítě respektující čas	10.168
Hustota sítě	0.074	Modularita	0.454
Prům. shluk. koeficient	0.4	Asortativita	0.034
Mocninný exponent	2.67		

Tabulka 15: Výsledky analýzy, Workplace

Časové okno	Počet uzlů	Počet hran	Průměrný stupeň	Komponent souvislosti	Největší komponenta	Hustota sítě	Mocninný exponent
1	46	53	2.304	3	41	0.051	2.43
2	31	29	1.871	6	9	0.062	4.47
3	53	90	3.396	3	48	0.065	2.47
4	38	30	1.579	9	16	0.043	2.4
5	29	25	1.724	7	13	0.062	2.78

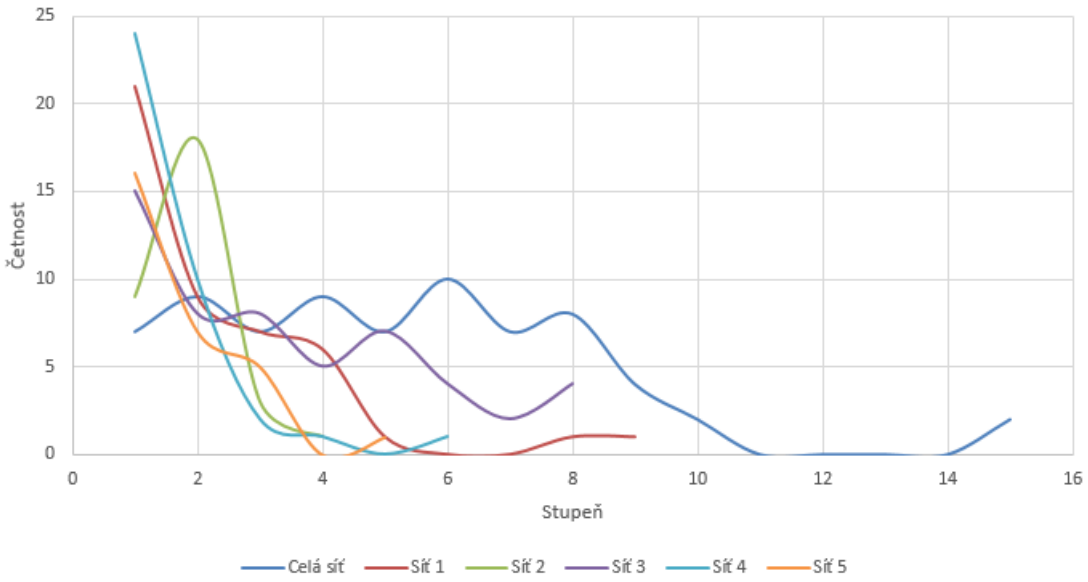
Tabulka 16: Výsledky analýzy časových oken, Workplace

Časové okno	Průměrná délka nejkratší cesty	Průměr sítě	Prům. shluk. koeficient	Modularita	Asortativita
1	4.498	11	0.102	0.063	-0.027
2	2.517	7	0.562	0.112	-0.042
3	3.393	8	0.579	0.116	0.0034
4	2.911	6	0.582	0.05	-0.052
5	2.979	8	0.584	0.273	0.086

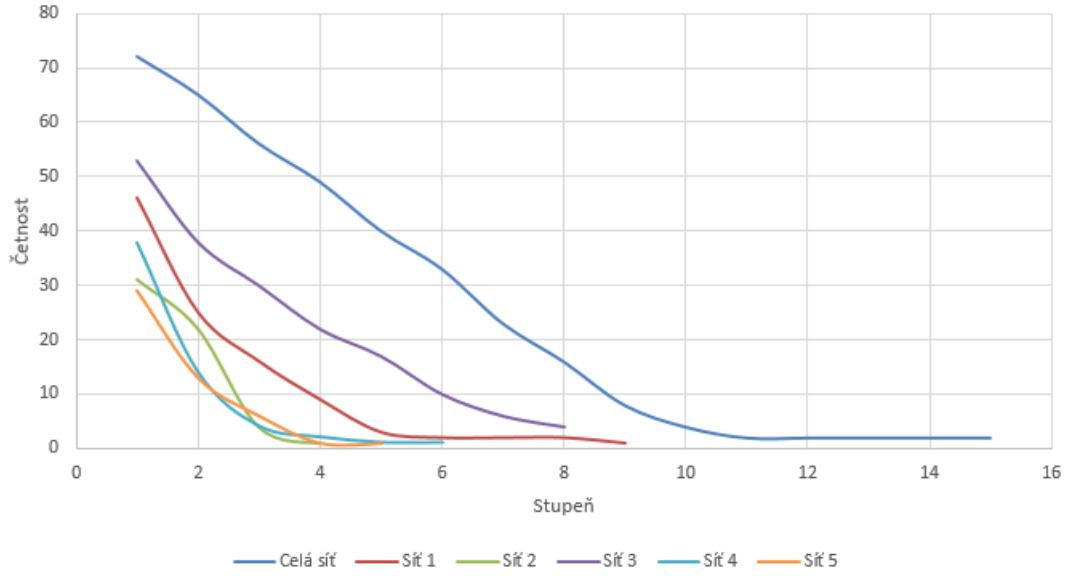
Tabulka 17: Výsledky analýzy časových oken, Workplace

Grafy distribucí

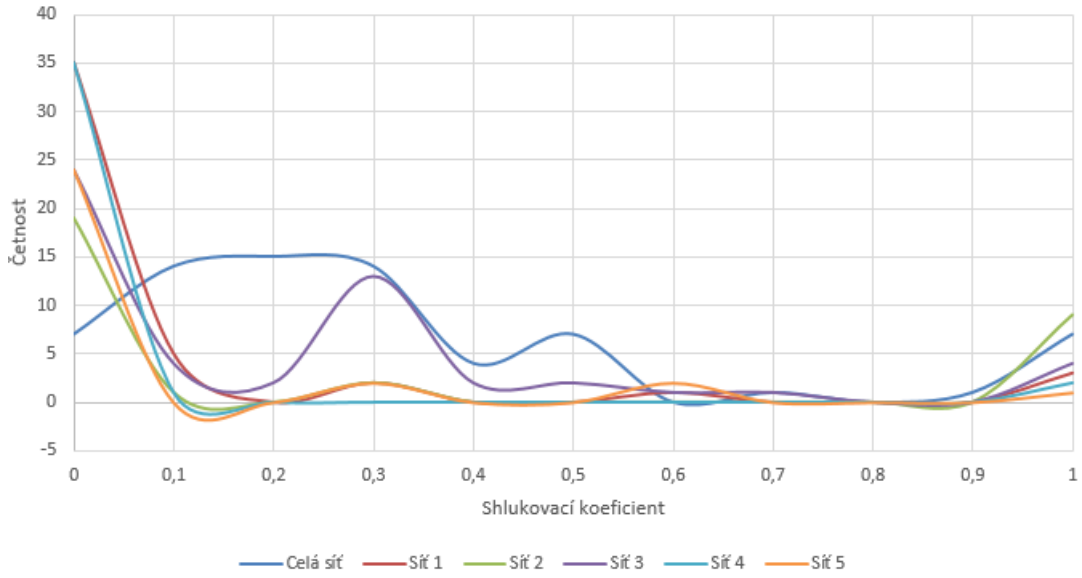
Na následujících obrázcích 38, 39 a 40 můžeme vidět distribuce pro sít Workplace. Pomocí kumulativní distribuce stupně jsem spočítal mocninný exponent.



Obrázek 38: Distribuce stupňů, Workplace



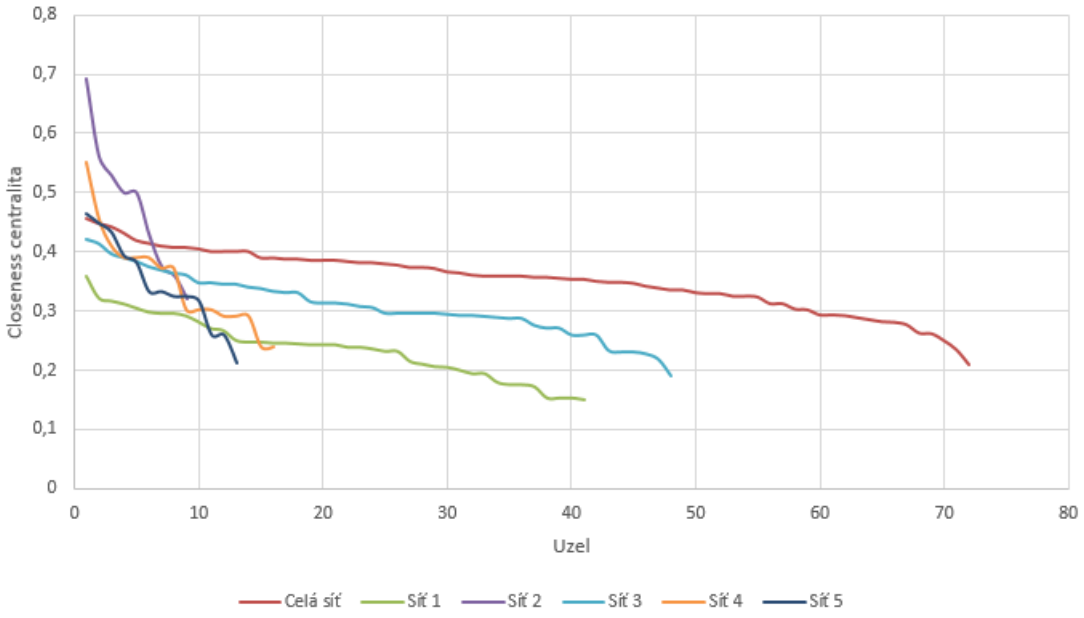
Obrázek 39: Kumulativní distribuce stupňů, Workplace



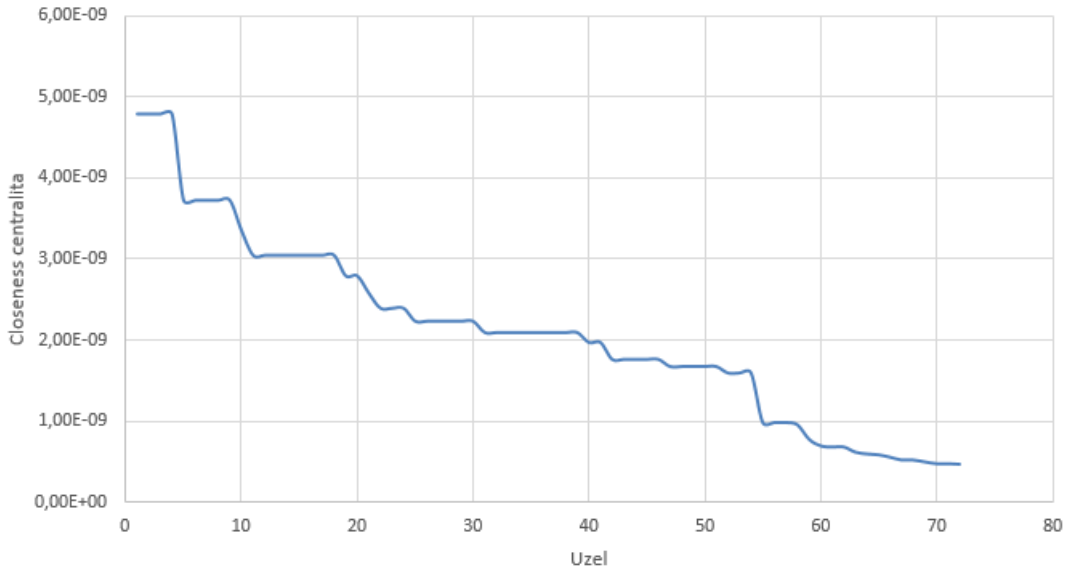
Obrázek 40: Distribuce shlukovacího koeficientu, Workplace

Closeness centralita

Na obrázku 41 jsou centrality pro celou síť i pro jednotlivá časová okna a a obrázku 42 je temporální closeness centralita.



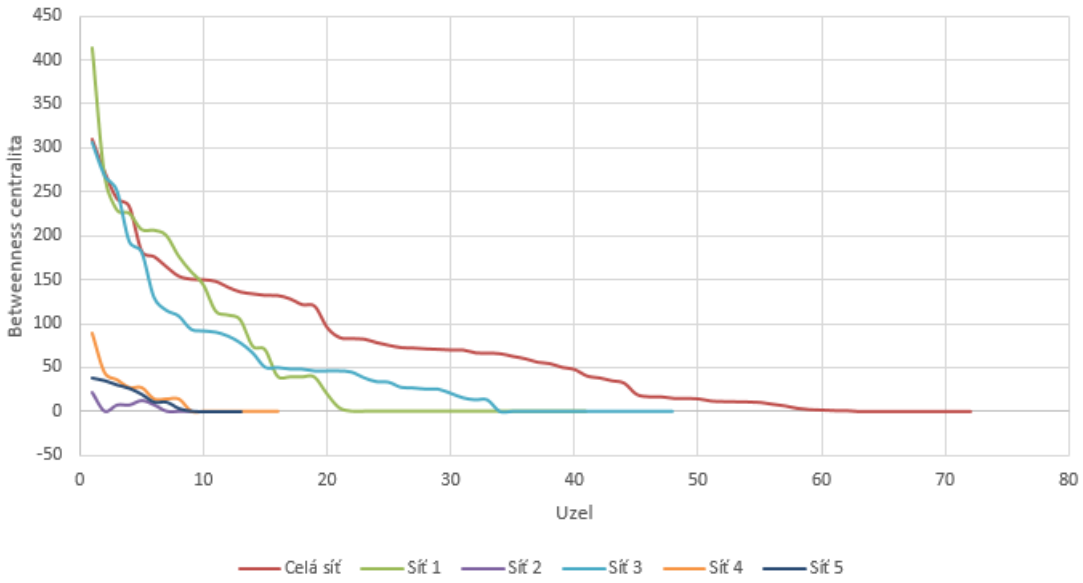
Obrázek 41: Closeness centralita, Workplace



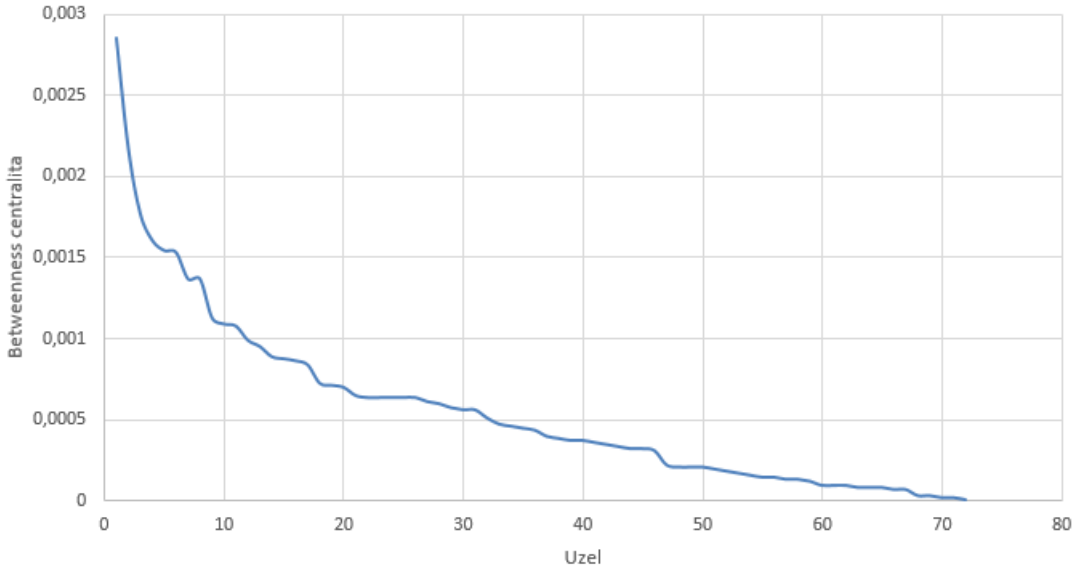
Obrázek 42: Temporální closeness centralita, Workplace

Betweenness centralita

Síť Workplace byla jediná u které se mi povedlo spočítat temporální betweenness centralitu, obrázek 43. Výsledek se moc neliší od netemporální betweenness centrality na obrázku 44.



Obrázek 43: Betweenness centralita, Workplace



Obrázek 44: Temporální betweenness centralita, Workplace

7 Závěr

Cílem práce bylo seznámit se s vybranými datovými kolekcemi temporálních sítí, provést nad nimi experimenty a vyhodnotit výsledky. Cílem taky bylo vytvořit vlastní aplikaci pro výpočet většiny vlastností temporálních sítí.

Implementovat vlastní aplikaci pro analýzu temporálních sítí nebyl velký problém, většina výpočtu nebyla příliš složitých. Problém nastal až při zpracování velkého množství dat, kde se jazyk Java ukázal jako ne moc vhodný. Práce s kolekcemi je sice jednoduchá, ale docela pomalá. Oproti nástrojům Gephi, nebo především R byly výpočty pomalejší. U malých sítí to nebyl velký problém, ale při analýze datové sady DBLP, už načítání a sestavení sítě trvalo poměrně dlouho. Gephi sice prováděl výpočty na středně velkých sítích rychleji, ale opravdu velké sítě už nedokázal zpracovat a celý program zamrzl. Nástroj R byl v tomhle tom vynikající a i ty největší sítě dokázal zpracovávat velmi rychle. Nakonec mi tento nástroj posloužil i pro kontrolu výsledků, Rozhodně se vyplatilo věnovat pozornost optimalizaci a ladění programu. Dokonce ve dvou případech se mi povedlo zrychlit program z několika hodin na několik sekund, jen tím, že jsem zvolil jiný přístup k prohledávání seznamů uzlů a hran, nebo si pomohl pomocnými seznamy. Obzvláště funkce pro hledání nejkratších cest respektujících čas vyžadovala mnoho přemýšlení.

Aplikace by se dala vylepšit lepším návrhem architektury. Optimalizací kódu a provádění některých výpočtů paralelně by se dala celá aplikace urychlit. Pro lepší práci s aplikací by se mohlo implementovat uživatelské rozhraní, místo konsolové aplikace.

Při experimentech se povedlo dosáhnout téměř všech cílů. Pouze temporální betweenness centralita byla spočítaná pouze pro jednu síť, protože hledání nejkratších cest respektujících čas bylo časově velmi náročné.

Literatura

[1] Petr Kovář. Úvod do teorie grafů, 2012. [online]. [cit. 2017-04-28]. Dostupné z: http://homel.vsb.cz/k̃ov16/files/uvod_do_teorie_grafu.pdf

[2] Petter Holme, Jari Saramäki. Temporal Networks, 2011. [online]. [cit. 2017-04-28]. Dostupné z: https://arxiv.org/pdf/1108.1780.pdf

[3] Jing Cui, Yi-Qing Zhang, Xiang Li. On the clustering coefficients of temporal networks and epidemic dynamics, 2013. [online]. [cit 2017-04-28]. Dostupné z https://www.researchgate.net/publication/261197069_On_the_clustering_coefficients_of_temporal_networks_and_epidemic_dynamics

[4] Vincent D. Blondel, Jean-Loup Guillaume, Renaud Lambiotte and Etienne Lefebvre. Fast unfolding of communities in large networks, 2008. [online]. [cit. 2017-04-28]. Dostupné z: https://arxiv.org/pdf/1108.1780.pdf

[5] M.E.J. Newman. Assortative mixing in networks, 2002. [online]. [cit 2017-04-28]. Dostupné z https://arxiv.org/pdf/cond-mat/0205405.pdf

[6] The R Project for Statistical Computing. [online]. [cit. 2017-04-28]. Dostupné z: https://www.r-project.org/

[7] DBLP. Computer Science Bibliography, 2017, [online]. [cit. 2017-04-28]. Dostupné z: http://dblp.uni-trier.de/statistics/

[8] SocioPatterns. High school contact and friendship networks, 2015, [online]. [cit. 2017-04-28]. Dostupné z: http://www.sociopatterns.org/datasets/high-school-contact-and-friendship-networks/

[9] Temporal networks. US Flights, 2015, [online]. [cit. 2017-04-28]. Dostupné z: http://vladowiki.fmf.uni-lj.si/doku.php?id=pajek:data:temp:air1

[10] SocioPatterns. Contacts in a workplace, 2016, [online]. [cit. 2017-04-28]. Dostupné z: http://www.sociopatterns.org/datasets/contacts-in-a-workplace/

A Příloha na CD

Součástí této práce je kompaktní disk, obsahující tento text v elektronické podobě, zdrojové kódy aplikace a soubory datových kolekcí. Kořenový adresář disku obsahuje následující podadresáře:

1. text - adresář obsahující tuto práci v PDF formátu
2. src - adresář se zdrojovými kódy aplikace
3. aplikace - adresář se spustitelnou aplikací
4. data - adresář s datovými kolekcemi